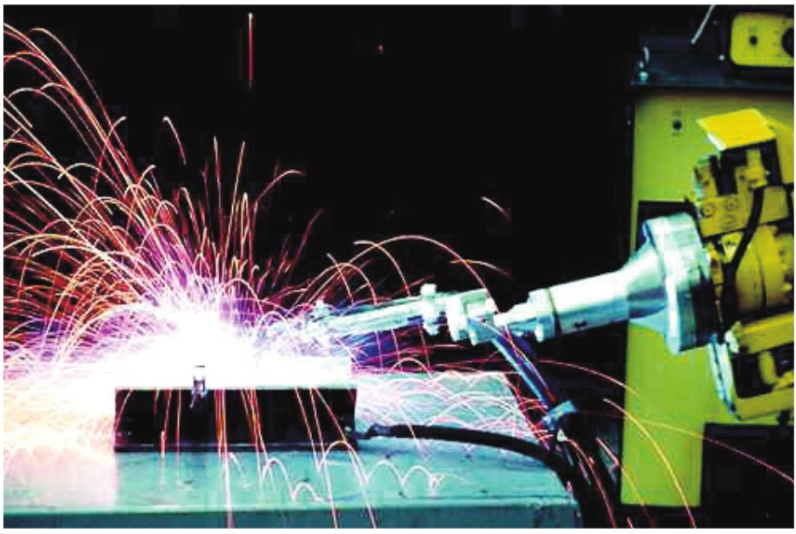
THROUGHPUT OPTIMIZATION IN ROBOTIC CELLS



Springer's INTERNATIONAL SERIES

MILIND W. DAWANDE
H. NEIL GEISMAR
SURESH P. SETHI
CHELLIAH SRISKANDARAJAH

Throughput Optimization in Robotic Cells

Recent titles in the **INTERNATIONAL SERIES IN OPERATIONS RESEARCH & MANAGEMENT SCIENCE**
Frederick S. Hillier, Series Editor, *Stanford University*

Gass & Assad/ *AN ANNOTATED TIMELINE OF OPERATIONS RESEARCH: An Informal History*
Greenberg/ *TUTORIALS ON EMERGING METHODOLOGIES AND APPLICATIONS IN OPERATIONS RESEARCH*
Weber/ *UNCERTAINTY IN THE ELECTRIC POWER INDUSTRY: Methods and Models for Decision Support*
Figueira, Greco & Ehrgott/ *MULTIPLE CRITERIA DECISION ANALYSIS: State of the Art Surveys*
Reveliotis/ *REAL-TIME MANAGEMENT OF RESOURCE ALLOCATIONS SYSTEMS: A Discrete Event Systems Approach*
Kall & Mayer/ *STOCHASTIC LINEAR PROGRAMMING: Models, Theory, and Computation*
Sethi, Yan & Zhang/ *INVENTORY AND SUPPLY CHAIN MANAGEMENT WITH FORECAST UPDATES*
Cox/ *QUANTITATIVE HEALTH RISK ANALYSIS METHODS: Modeling the Human Health Impacts of Antibiotics Used in Food Animals*
Ching & Ng/ *MARKOV CHAINS: Models, Algorithms and Applications*
Li & Sun/ *NONLINEAR INTEGER PROGRAMMING*
Kaliszewski/ *SOFT COMPUTING FOR COMPLEX MULTIPLE CRITERIA DECISION MAKING*
Bouyssou et al./ *EVALUATION AND DECISION MODELS WITH MULTIPLE CRITERIA: Stepping stones for the analyst*
Blecker & Friedrich/ *MASS CUSTOMIZATION: Challenges and Solutions*
Appa, Pitsoulis & Williams/*HANDBOOK ON MODELLING FOR DISCRETE OPTIMIZATION*
Herrmann/*HANDBOOK OF PRODUCTION SCHEDULING*
Axsäter/ *INVENTORY CONTROL, 2nd Ed.*
Hall/ *PATIENT FLOW: Reducing Delay in Healthcare Delivery*
Józefowska & Węglarz/ *PERSPECTIVES IN MODERN PROJECT SCHEDULING*
Tian & Zhang/ *VACATION QUEUEING MODELS: Theory and Applications*
Yan, Yin & Zhang/ *STOCHASTIC PROCESSES, OPTIMIZATION, AND CONTROL THEORY APPLICATIONS IN FINANCIAL ENGINEERING, QUEUEING NETWORKS, AND MANUFACTURING SYSTEMS*
Saaty & Vargas/ *DECISION MAKING WITH THE ANALYTIC NETWORK PROCESS: Economic, Political, Social & Technological Applications w. Benefits, Opportunities, Costs & Risks*
Yu/ *TECHNOLOGY PORTFOLIO PLANNING AND MANAGEMENT: Practical Concepts and Tools*
Kandiller/ *PRINCIPLES OF MATHEMATICS IN OPERATIONS RESEARCH*
Lee & Lee/ *BUILDING SUPPLY CHAIN EXCELLENCE IN EMERGING ECONOMIES*
Weintraub/ *MANAGEMENT OF NATURAL RESOURCES: A Handbook of Operations Research Models, Algorithms, and Implementations*
Hooker/ *INTEGRATED METHODS FOR OPTIMIZATION*

- ***A list of the early publications in the series is at the end of the book ****

Cover Photo: Purchased from Getty Images
Cover Design: Anjuli Sethi

THROUGHPUT OPTIMIZATION IN ROBOTIC CELLS

MILIND W. DAWANDE
School of Management, University of Texas at Dallas

H. NEIL GEISMAR
College of Business, Prairie View A&M University

SURESH P. SETHI
School of Management, University of Texas at Dallas

CHELLIAH SRISKANDARAJAH
School of Management, University of Texas at Dallas

Springer

Milind W. Dawande
University of Texas at Dallas, USA

H. Neil Geismar
Prairie View A&M University, USA

Suresh P. Sethi
University of Texas at Dallas, USA

Chelliah Sriskandarajah
University of Texas at Dallas, USA

Library of Congress Control Number: 2007920608

ISBN-10: 0-387-70987-8 (HB) ISBN-10: 0-387-70988-6 (e-book)
ISBN-13: 978-0387-70987-1 (HB) ISBN-13: 978-0387-70988-8 (e-book)

Printed on acid-free paper.

9 8 7 6 5 4 3 2 1

springer.com

To my parents Wasudeo and Arundhati. *Milind Dawande*

To Karen. *Neil Geismar*

To the memories of my mother Manak and aunt Manori.
To my sisters Mena, Kamalsri, and Sulochana.
To my sisters-in-law Sohani, Tara, Amrao, and Mena.
To my wife Andrea and my daughters Chantal and Anjuli. Suresh Sethi

To my wife Kohila.
To my children Vani and Madan. *Chelliah Sriskandarajah*

Foreword

This book presents research on sequencing and scheduling problems in robotic cells. The authors have been at the forefront of research activity in this area. Suresh Sethi and Chelliah Sriskandarajah coauthored (together with Gerhard Sorger, Jacek Błażewicz, and Wieslaw Kubiak) an influential paper titled "Sequencing of Parts and Robot Moves in a Robotic Cell," (*International Journal of Flexible Manufacturing Systems*, 1992) that helped establish the framework for the algorithmic investigation of throughput optimization problems in the robotic cell literature. Along with their colleague Milind Dawande and former student Neil Geismar, they have put together this treatise that incorporates their own research and that of others.

The authors have done a commendable job in bringing together the important analytical results on throughput optimization in a variety of robotic cells. The book starts by providing the reader with a snapshot of the different applications of robotic cells in the industry. In particular, such cells are used extensively in the production of semiconductors. The authors then devise a classification scheme (Chapter 2) for the scheduling problems that arise in the different types of robotic cells. Cyclic production, the most commonly used mode of production, is analyzed next (Chapter 3). Using a basic model of a robotic cell, the authors explain the notion of cycles and cycle times, and proceed to derive a variety of results, exact and approximation algorithms, concerning cyclic production. Scheduling problems in cells with more advanced hardware are discussed next. In Chapter 4, algorithms are presented for cells in which the robot has a gripper that can hold two parts simultaneously.

Chapter 5 discusses cells that have more than one machine at one or more processing stages. In Chapters 6 and 7, the authors then widen the scope of inquiry by addressing cells which are able to produce two or more different types of parts simultaneously. Cells with more than one robot are discussed in Chapter 8. Most of the descriptions in Chapters 3-8 are for cells in which a part that has completed processing on a given machine can stay on that machine indefinitely (until a robot picks it up). Chapter 9 briefly discusses two other types of cells that have noteworthy practical applications. The final chapter (Chapter 10) presents a number of open problems.

Throughput optimization problems for robotic cells are not at all like the classical machine scheduling problems with which I am familiar with and have published papers on. The notation required to state and analyze these problems is significantly different than that used in the previous scheduling literature, and it may take a reader some effort to gain familiarity with the notation used in this book. However, the reader will find the effort worthwhile and will appreciate that this new area has a number of well-defined and non-trivial combinatorial problems stemming from practical applications. Efficient solution techniques for solving such problems may lead to significant cost savings for factories using robotic cells in their production processes.

Two of the authors of this book, Suresh Sethi and Milind Dawande, received their Ph.D. degrees from the Graduate School of Industrial Administration (now the Tepper School of Business) at Carnegie Mellon University. Suresh was the first student to complete his Ph.D. degree in five semesters under my direction during the 43 years that I was a faculty member of Carnegie Mellon University.

Pittsburgh, Pennsylvania
August 2006

Gerald L. Thompson
Professor of Operations Research, Emeritus
Carnegie Mellon University

Contents

Preface

The love of learning, the sequestered nooks,
And all the sweet serenity of books.
– Henry Wadsworth Longfellow, 'Morituri Salutamus,' 1875

Intense global competition in manufacturing has compelled manufacturers to incorporate automation and repetitive processing for improving productivity. As manufacturers strive to reduce cost and improve productivity, the benefits offered by computer-controlled material handling systems – efficiency and speed of processing, reduced labor costs, a largely contaminant-free environment, to name a few – are compelling reasons for their use. In their typical use, such systems are responsible for all inter-machine handling of work-in-process as raw materials progress through the multiple processing stages required to produce a finished part.

Many modern manufacturing systems use robot-served manufacturing cells, or *robotic cells* – a particular type of computer-controlled manufacturing system in cellular manufacturing. The exact time of the first use of such systems is difficult to pinpoint; however, several industrial implementations were in use in the 1970s. Most of these were classical machining applications such as automated tool loading and unloading for metal-cutting, grinding, turning, etc., and automated classification of parts before palletizing. Over the years, the scope has broadened to a wide variety of industries including the manufacture of semiconductors, textiles, pharmaceutical compounds, magnetic resonance imaging systems, glass products, cosmetics, fiber-optics, and building products.

As they become prevalent, using robotic cells efficiently becomes a competitive necessity. In this regard, research efforts have focused on three major issues: cell design, sequencing of robot moves, and optimal scheduling of the parts to be produced. The latter two issues are the subject of most of our discussion in this book. In particular, our emphasis is on cyclic production in which a sequence of robot actions is repeated until the production target is met.

This book is devoted to consolidating the available structural results about cyclic production in the various models used to represent real-world cells. As cells become larger and more complex, the need for increasingly versatile models and easy-to-implement algorithms to optimize cell operations has increased. We have made an attempt to bring together the results developed over the past 25 years. The material is organized into 10 chapters. We start by taking a look at industrial applications and formulating a classification scheme for robotic cell problems. After presenting some fundamental results about cyclic production, we proceed to analyze cells with dual-gripper robots, parallel machines, multiple-part-type production, and multiple robots. Finally, we discuss some important open problems in the area.

We envision this book as a reference resource for practitioners, researchers, and students. The book can also be used in a graduate course or a research seminar on robotic cells.

We extend our grateful thanks to our numerous colleagues whose contributions have been directly or indirectly included in this book. In particular, we are indebted to our co-authors, Jacek Błażewicz, Inna Drobouchevitch, Nicholas Hall, Hichem Kamoun, Wieslaw Kubiak, Subodha Kumar, Rasaratnam Logendran, Chris Potts, Natarajan Ramanan, Jeffrey Sidney, and Gerhard Sorger, whose collaboration has been critical for the development of a significant portion of the material covered in the book. We thank Alessandro Agnetis, Nadia Brauner, Chengbin Chu, and Eugene Levner for their encouragement and for suggesting several improvements to the manuscript. We also thank our student Mili Mehrotra for her help in proofreading parts of the manuscript. It was a pleasure working with Gary Folven and Carolyn Ford of Springer; we are grateful for their support. Finally, we thank Barbara Gordon for her help with LaTeX.

M.W. Dawande
S.P. Sethi
C. Sriskandarajah
Richardson, TX

H.N. Geismar
Prairie View, TX

Chapter 1

ROBOTIC CELLS IN PRACTICE

Computer-controlled material handling systems that convey raw materials through the multiple processing stages required to produce a finished part or product are common in industry. One such implementation, a robotic cell, has become a standard tool in modern manufacturing. The efficient use of such cells requires algorithmic solutions to a variety of challenging combinatorial optimization problems; typical problems include cell design, optimal sequencing of robot moves, and scheduling of the products to be produced.

Many diverse industries use robotic cells (see Section 1.5). A dominant area of application is semiconductor manufacturing [6, 64, 102, 128, 129, 154, 159]. Other documented implementations include electroplating lines for a variety of products ranging from printed circuit boards to airplane wings, where parts are transferred between tanks of chemicals by hoists [30, 31, 33, 97, 98, 108, 109, 114, 145]. Robotic cells are also used for testing and inspecting boards used in mainframe computers [121], machining of castings for truck differential assemblies [8], crane scheduling for computer integrated manufacturing, textile mills, and engine block manufacturing [149].

As manufacturers implement larger and more complex robotic cells, more sophisticated models and algorithms are required to optimize the operations of these cells. A number of studies have been conducted to find ways to meet this demand. Some date as far back as the late 1970s, but the majority have been performed since 1990. This chapter pro-

vides a general introduction to the use of robotic cells in practice. The concept of cellular manufacturing is discussed in Section 1.1, followed in Section 1.2 by the basic robotic cell flowshop models useful for analyzing industrial implementations. Section 1.3 discusses the importance of schedule optimization. Section 1.4 provides a brief historical overview of robotic cell studies. Finally, Section 1.5 provides a snapshot of the use of robotic cells in industry.

1.1 Cellular Manufacturing

The idea of using robotic cells for production is part of the larger theme of *cellular manufacturing*. The origin of cellular manufacturing can be traced back to efforts aimed at blending the efficiency of product layouts (e.g., assembly lines) with the flexibility of process or functional layouts (e.g., job shops). These layouts represent two extreme paradigms in manufacturing: on the one hand, process layouts are characterized by the utilization of general-purpose resources to produce a variety of products. Typically, the processing requirements vary widely with the products; resources are grouped by similarities in their processing activities. Advantages of process layouts include the ability to handle a large variety of products, thereby enabling customization, and the use of general-purpose equipment. On the other hand, product layouts operationalize the idea of using specialized resources for producing a few standardized goods. Such layouts achieve processing precision and speed through the specialization of labor and equipment. A cellular layout is an attempt to exploit the advantages of both of these extremes.

Production cells may be formed in a job shop by grouping together machines that perform different operations in order to produce a set of items, or *part family*, that requires similar processing. For example, a lathe, a drill press, a milling unit, and a grinder in a machine shop may each be pulled from their respective work centers and configured into a cell to produce several batches of related parts. This arrangement may be either temporary (i.e., to serve a particular customer order) or permanent. A permanent work cell is also referred to as a *focused work center* (Heizer and Render [82]).

A cellular layout is generally preferred to a product layout if the scale of production, either in volume or number of operations, is not very large.

Cells require less floor space, less initial investment in machinery and equipment, and less raw material and finished goods inventory than do product layouts. The advantages of cellular over process layouts include faster throughput, reduced setup time, less work-in-process inventory, reduced direct labor costs, and increased utilization of machinery and equipment (Stevenson [148]). Simulation studies (e.g., Sassani [140]) have supported these assertions by finding that a cellular layout works most efficiently if the products for each cell are well defined and the cells are isolated. The smoothness of production deteriorates as the product mix and product design become more varied. However, efficiency can be regained while serving this broadening set of requirements by developing additional cells to handle them.

For cellular manufacturing to be effective, there must be families of items that have similar processing characteristics. Moreover, these items must be identified. The process that organizes the products is known as *group technology*, and commonly involves identifying items that have similarities in design characteristics (size, shape, and function) and then classifying them into part families via some coding scheme. Parts of a given family are then processed in a specific cell. An alternative approach to grouping families of parts is to differentiate based on manufacturing characteristics (type and sequence of operations required); this approach is commonly referred to as *production flow analysis* (Burbidge [26, 27]). However, some have noted the difficulties in implementing this method for a facility with a large number of parts (Nahmias [124]). A more thorough discussion of coding systems and how they relate to group technology can be found in Groover and Zimmers [72].

1.2 Robotic Cell Flowshops

This book focuses on sequencing and scheduling for a particular type of automated material handling system in cellular manufacturing: robotic cells. A robotic cell consists of an input device; a series of processing stages, each of which performs a different operation on each part in a fixed sequence; an output device; and one or more robots that transport the parts within the cell. Each stage has one or more machines that perform the processing for that stage. The default configuration of a robotic cell (Figure 1.1) is to have one robot that can hold only

one part, one machine per stage, and no buffers for intermediate storage between the stages within the cell; variations to this configuration (e.g., Figure 1.2) are presented throughout the book.[1] Because each machine can hold only one part, a robotic cell is, in essence, a flowshop with blocking (Pinedo [132]) that has common servers that perform all transfers of materials between processing stations. In robotic cells, each function

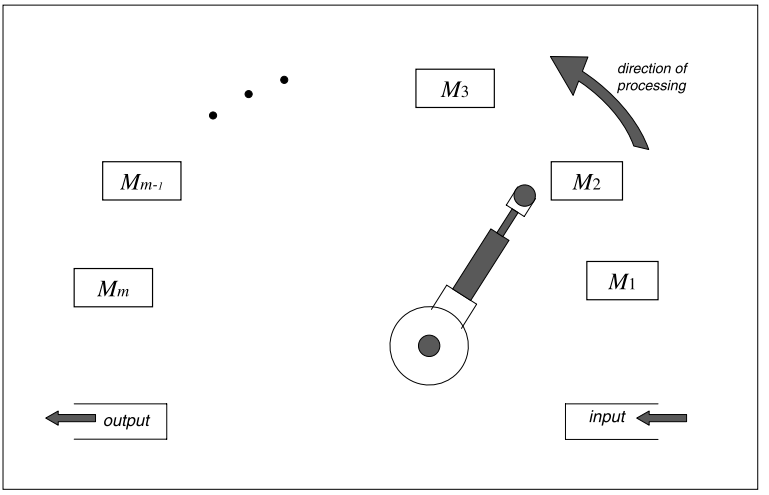


Figure 1.1. A Generic Robot-Centered Cell.

in the process is performed by a machine; there are no human-tended workstations. Because the material handling – movement between machines and loading/unloading of machines – is performed by robots, a variety of *remote center compliance* devices are available to ensure that the parts can be reliably loaded onto the machines in the correct orientation (see Groover et al. [71] for a description of these devices). Once this is accomplished, robots are advantageous for material handling because they can operate with speed and precision consistently for long periods. In addition, in some environments they are preferred because their use prevents contamination; examples include pharmaceutical compounding

[1] Parallel machines (Chapter 5), dual-gripper robots (Chapter 4), multiple robots (Chapter 8), and internal buffers (Chapter 4).

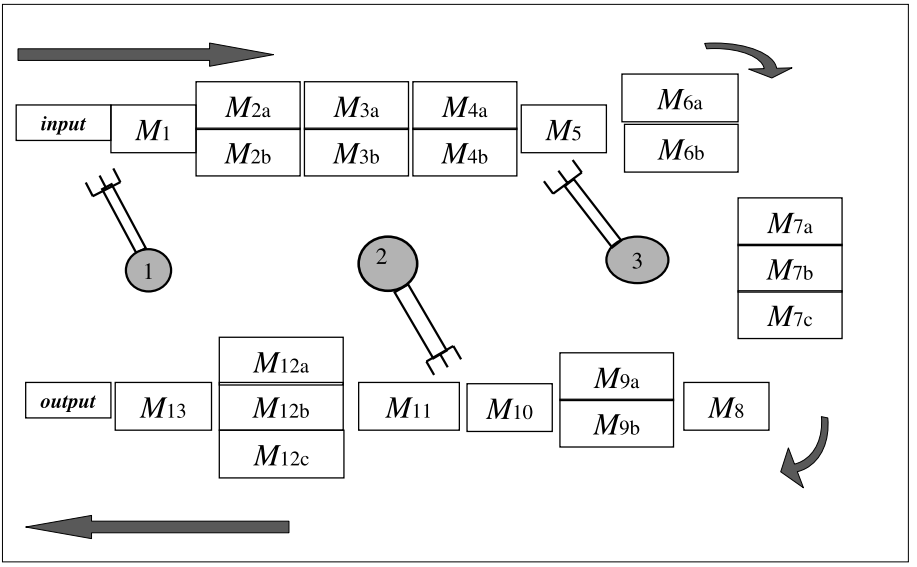


Figure 1.2. A Robotic Cell with Parallel Machines and Multiple Robots with Dual Grippers.

and semiconductor manufacturing. Other manufacturing environments are inhospitable to humans, so robots are a natural alternative: some semiconductor processing is done in a vacuum; welding and iron-working applications may be in high-temperature environments; and painting or applying other types of coatings may emit noxious fumes.

The functions performed by the machines obviously depend on the use of the cell. In semiconductor photolithography (Kumar et al. [102]), the operations include bake, chill, coat, expose, develop, and scan. A cell designed for machining 23 different components of a valve performs milling, drilling, roughing and finishing bores, chamfering, reaming, spot-facing, deburring, and threading (Bolz [16]). A cell that processes large investment castings may perform operations such as slurry dip and sand coating, and one that produces agricultural equipment uses chuckers, shapers, and shavers (Kafrissen and Stephans [88]). IBM's Poughkeepsie, NY, plant has used a robotic cell to write and package diskettes with operations including diskette writing, envelope printing, and la-

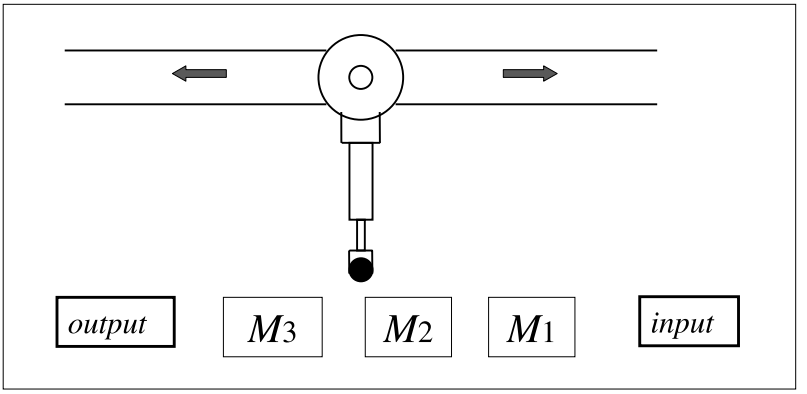


Figure 1.3. A Three-Machine Overhead Track-Based Robot Cell.

bel printing (Giacobbe [66]). One example of a cell designed for automated chemical analysis includes a spectrophotometer, a continuous dilution apparatus, an electronic balance, a magnetic hotplate-stirrer, and a pump for dispensing solvents (Dunkerley and Adams [53]).

Different types of robots are used in industrial applications. In a common implementation for semiconductor manufacturing, the robot has a fixed base and an arm that rotates, as in Figure 1.1. Such a cell is commonly called a robot-centered cell. In another configuration, often used in electroplating printed circuit boards, the robot is attached by a hoist to an overhead track, and the entire robot moves linearly along this track (see Figure 1.3). A more general case combines these two: the robot's arm rotates on its base, and the robot itself moves linearly along a track (see Figure 1.4). This configuration is called a mobile robot cell. Obviously, such a system allows the robot to cover a larger area. The cell layouts themselves fall into two basic categories. The first, demonstrated in Figures 1.1, 1.3, and 1.4, can be either linear or semicircular. A significant characteristic of this category is that in order to travel from the output buffer to the input buffer, the robot must pass by each of the machines. Compare this to the cell in Figure 1.5(a). In this cell, the robot may travel from the output buffer to the input buffer directly. In addition, it may travel from the output buffer to machine M_1

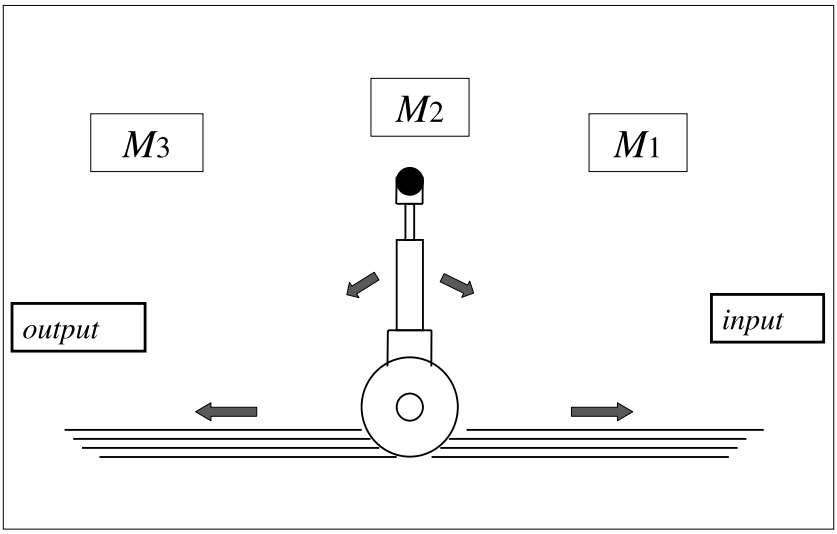


Figure 1.4. A Cell with More General Robot Capabilities: The Robot's Arm Rotates on Its Base, and the Robot Itself Moves Linearly Along a Track.

by traveling either clockwise (passing machines M_2 and M_3) or counterclockwise (passing the input buffer). This flexibility significantly impacts the sequence of robot moves which results in optimal productivity from the cell. A circular cell in which the input buffer and output buffer share the same location is shown in Figure 1.5(b).

1.3 Throughput Optimization

Standardization of the processing requirements of the parts or products together with the volume required creates an ideal environment for repetitive production. In their typical use in practice, robotic cells are employed to produce significant volumes of either a single part or a few closely related parts. Given the processing requirements, the objective that most interests manufacturers is the maximization of cell productivity. A natural and widely used measure of productivity is *throughput* – the number of finished parts produced per unit of time.

Given the goal of maximizing the throughput of the cell, two remarks need immediate mention. One, small improvements in throughput can improve revenues significantly for one or both of the following reasons: the significant volumes produced by the cell and the high market value of the products. A case in point is semiconductor wafer fabrication,

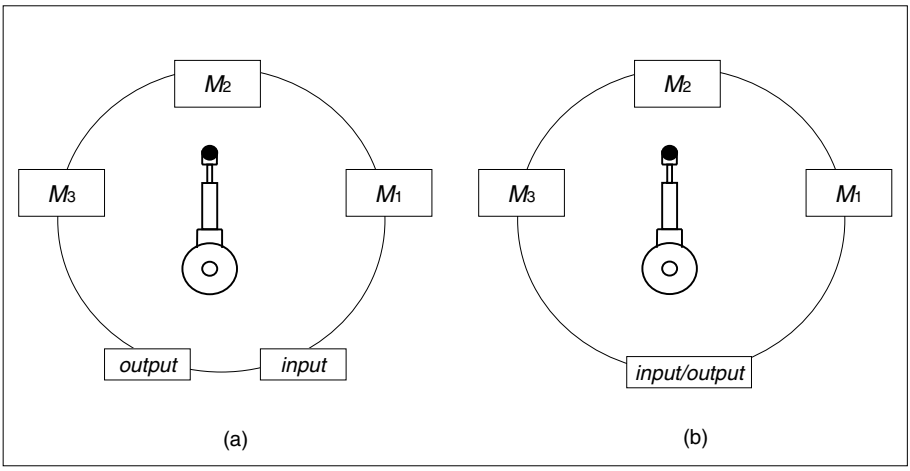


Figure 1.5. (a) A Three-Machine Robot Cell, (b) A Three-Machine Robot Cell with Input and Output Together.

where both of these factors contribute to focus attention on throughput maximization. For example, Geismar et al. [64] show how an 8% increase in throughput can increase a semiconductor manufacturer's revenues by almost $3 million per week. Second, a number of cell characteristics impact its throughput. These include processing speeds of the machines and robots, cell layout, and the sequence of robot actions. For a specific manufacturing environment, an *a priori* judgment about the relative impact of these characteristics is often difficult to ascertain.

Over the years, with an increase in the size and the processing complexity of robotic cells, optimization of the schedule of robot moves has emerged as a dominant tool for achieving throughput maximization. In practice, many cell parameters are fixed by physical constraints and cannot be altered. There is generally little flexibility in the layout of the cell, and changes to it would have relatively less influence on throughput. In most applications, processing requirements are strict; reducing the processing time at a stage would change the nature of that operation and its result. The processing speeds at the different stages of the

cell are constrained by the latest technology available. Although various robots with different speeds are available, once a robot is purchased, it comes with a specified processing speed that cannot be changed.

Alternatively, the best schedule of robot moves can be chosen relatively inexpensively. This requires no changes to the layout, the machines, or the robots. Many of the results can be proven analytically or demonstrated through simulation, rather than trying different schedules in production. Consequently, implementing a schedule change requires little nonproductive time for the cell. Furthermore, these tactics can be used to determine the benefits that can be realized by adding advanced hardware, such as parallel machines or dual-gripper robots, as shown in Chapters 4 and 5.

1.4 Historical Overview

Because of the nature of processing requirements, the theoretical underpinnings of throughput optimization in robotic cells are in the area of flowshop scheduling. During the past 50 years, a huge body of literature has analyzed a variety of flowshop operations. A number of books and surveys (see e.g., Brucker [25], Lawler et al. [104], Pinedo [132]) discuss developments in this area; we, therefore, mention just a few works that are relevant to our discussion. A classical result for optimal job scheduling in a two-machine flowshop is derived by Johnson [86]. Wagner [155] formulates an integer program for an m-machine flowshop. Garey et al. [58] establish the NP-hardness of the m-machine flowshop problem for $m \geq 3$. Abadi et al. [1], Levner [110], and Pinedo [131] study flowshops with blocking. Papadimitriou and Kannelakis [127], Reddi and Ramamoorthy [135], Röck [136], Sahni and Cho [139], and Wismer [157] are early works on no-wait flowshops.

In one of the early papers on robotic cell sequencing, Bedini et al. [11] develop heuristic procedures for optimizing the working cycle of an industrial robot equipped with two independent arms. Baumann et al. [10] derive models to determine robot and machine utilization. Maimon and Nof [118] and Nof and Hannah [126] study cells with multiple robots. Devedzic [48] proposes a knowledge-based system to control the robot.

Wilhelm [156] classifies the computational complexity of a number of scheduling problems in assembly cells. In a later study, Hall and

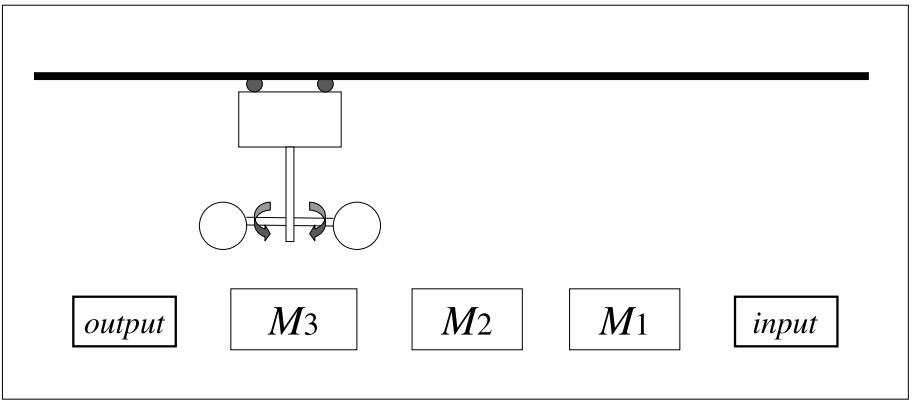


Figure 1.6. A Dual-Gripper Overhead-Track Cell.

Sriskandarajah [77] survey scheduling problems with blocking and no-wait conditions and classify their computational complexities.

Early studies use simulation to compute cycle times. Kondoleon [100] uses computer modeling to simulate robot motions in order to analyze the effects of configurations on the cycle time. Claybourne [35] performs a simulation to study the effects that sequencing robot actions has on throughput. Asfahl [8] simulates the actions of a robotic cell with three machines to demonstrate the transition from cold start to steady-state cyclic operations. Błażewicz et al. [15] develop an analytical method to derive cycle time expressions for robotic cells. Dixon and Hill [49] compute cycle times by using a database language to simulate robotic cells.

Sethi et al. [142] set the agenda for most subsequent studies on cells. They provide analytical solutions to the problem of sequencing robot moves for two-machine and three-machine cells that produce identical parts, and for two-machine cells that produce different parts. Logendran and Sriskandarajah [116] generalize this work to cells with different types of robots and with more general robot travel times. Brauner and

Finke [18, 20, 21, 22] perform several studies that compare 1-unit cycles with multi-unit cycles (cycles are defined in Chapter 3). Crama and van de Klundert [40] develop a polynomial-time algorithm for finding an optimal 1-unit cycle in an additive travel-time cell (travel times are defined in Section 2.2.2). Dawande et al. [47] do the same for constant-travel-time cells. Brauner et al. [24] prove that finding an optimal robot move sequence in a robotic cell with general travel times is NP-hard. Hall et al. [75, 76] and Sriskandarajah et al. [147] study part scheduling problems and their complexities for cells that process parts of different types. Geismar et al. address robotic cells with parallel machines [59] and with multiple robots [64]. There have also been several studies [50, 51, 61, 143, 146, 149, 154] on robotic cells with dual-gripper robots (defined in Chapter 4; see Figure 1.6).

Research on cells with no-wait or interval pickup has been performed in parallel with the above-mentioned studies on free-pickup cells (see Section 2.2.1 for information on pickup criteria). Levner et al. [111] develop an algorithm for an optimal 1-unit cycle in a no-wait cell that produces identical parts. Agnetis [2] finds optimal part schedules for no-wait cells with two or three machines. Agnetis and Pacciarelli [3] study the complexity of the part scheduling problem for three-machine cells. Che et al. [31] present a polynomial-time algorithm for an optimal 2-unit cycle in no-wait cells that produce identical parts or two part-types. Kats and Levner [97] address no-wait cells with multiple robots.

An early work on interval robotic cells is that by Lei and Wang [108], who use a branch-and-bound search process. Chen et al. [33] use branch-and-bound, linear programming, and bi-valued graphs to find optimal 1-unit cycles, and Che et al. [30] employ these techniques to find optimal multi-unit cycles. Kats et al. [98] solve this problem using a method similar to that used by Levner et al. [111] for no-wait cells. Complexity results for such a system are presented in Crama [38], Crama and van de Klundert [41], and van de Klundert [153].

1.5 Applications

This section presents several examples of how robotic cells are used in the industry, discusses their capabilities, and addresses issues in their implementation.

Robotic cells often perform the general functions of arc welding, material handling, and machining [161]. In addition to semiconductor manufacturing, electroplating, and textiles, they are used in many different industries, including injection molding of battery components [173], glass manufacturing and processing [162], building products [165], cosmetics [166], lawn tractors [167], and fiber-optics [174]. In the medical field, robotic cells are used to produce components for magnetic resonance imaging systems [163], for automated pharmacy compounding [168], to process nucleic acids, and to generate compounds for tests in relevant biological screens (Rudge [138]). Cells for grinding, polishing, and buffing handle many products, including rotors, stainless steel elbows for the chemical and the food industries, sink levers and faucets, propane tanks, flatware, and automotive products [171].

The Rolls Royce aircraft engine plant in Derby, England, uses a sequence of seven robotic cells to machine jet engine turbine blades. These turbine blades must be produced to extremely high quality standards, so the cells support advanced casting techniques and blade materials. The automated line, which uses creep-feed grinding rather than milling or broaching, has increased throughput from ten per hour to eighty per hour. This improvement has allowed Rolls Royce to change from machining batches of blades to producing individual blades, thereby improving manufacturing flexibility and reducing lead time and inventory (Bolz [16]).

The Sperry Vickers plant in Omaha, Nebraska, uses two robotic cells to machine 28 varieties of hydraulic pump cover castings made of ductile cast iron. Each cell has a Unimate robot that can handle up to 112.5 kg and has five axes of freedom. After the machining operations (milling, rough and finish boring, drilling, and facing), the castings are washed and then checked for quality by an automated gage. The robot then places a casting onto an output conveyor or the reject conveyor, depending on the result of the gaging operation (Maiette [117]).

The Flexible Automated Manufacturing Technology Evaluation Center at the Illinois Institute of Technology designed a cell that uses a laser at one stage. This cell produces process control valves that are used to regulate the flow or pressure of a variety of gases and liquids. Plugs are machined from different materials (e.g., steel, aluminum, brass), depending on the fluid to be regulated. The operations in the cell are performed

by a CNC lathe, a hardening station at which the laser is used for heat treatment or cladding, an inspection station, and a cleaning station that uses slotted-paper polishing wheels. A completed plug is produced every eight minutes (Sciaky [141]).

Miller and Walker [122] describe several real-world implementations of robotic cells. One example is that of a robot-centered cell with four machines – an NC lathe, a surface grinding machine, and two NC drilling machines – serviced by a single-gripper robot. Another example describes the use of a dual-gripper robot for producing a family of duplicator fuser rollers at Xerox Corporation.

In modern manufacturing, a typical robot may move along six axes (including linear translation) and have a three-fingered pneumatic gripper [164]. Some have angular and parallel motion grippers that include miniature, low-profile, sealed, long jaw travel, and 180 degree jaw motion grippers [160]. Robots that can calculate the optimal path between two locations or that can quickly change their tools are common [161]. Robotic vision-guided systems have grown in the market, especially for assembly cells [169].

The economic benefits of robotic cells extend beyond increasing the efficiency of manufacturing. One company states that its 19 cells will achieve their payoff mark in only 2 years [173]. Another notes that implementing robotic cells has consolidated several processes, which has reduced floor space requirements [165]. Such successes have helped the robotic cell market grow at a healthy rate for the past few years [161, 170].

Companies typically use simulators to study their robotic cells because factories are often too large, too complex, and too costly to be optimized any other way. In addition, there are currently no analytical models that accurately depict cells with general travel times, stochastic processing times, or random machine failures. Some simulators claim to model automated systems with better than 98% accuracy (Fowler et al. [55]). Among the topics studied via simulators are the influence of adding a parallel machine to a bottleneck process, the effects of equipment failures and maintenance on performance, and the disruption caused by introducing high-priority jobs into steady-state production lines (Duenyas et al. [52]). Simulators are also used for sensitivity analysis to determine if equipment purchases are required to meet new production goals (Fowler

et al. [55]). Because simulators can predict and verify throughput before a cell is assembled, they can be used to improve the cell's layout during the planning stage. This capability can also influence product design by leading to changes that make the product's fabrication more efficient in an existing cell [172].

For cells with stochastic data, some companies implement feedback control by using dispatching rules that determine the robot's next move based on the current state of the cell. One such rule is called Longest Waiting Pair (see Chapter 8). To implement this scheme, the control computer tracks each part whose processing has completed on its current machine and is waiting to be moved. It also tracks each empty machine that is waiting for the next part to process. Each part's waiting time is summed with the waiting time of the machine to which it travels next. For the pair with the largest combined waiting time, the robot's next move is to carry the part to its corresponding machine (Kumar et al. [102]).

When designing a cell and its operating parameters, the main objective is the maximization of the cell's throughput. Intermediate goals toward reaching this objective are high machine utilization and a smooth distribution of work over the entire system. Management must balance the pursuit of these goals with its desire to reduce work-in-process inventory levels. As the system operates, bottleneck identification and knowing which lots might be late become important objectives (Duenyas et al. [52]). General guidelines for applying operations research techniques to planning, designing, and operating robotic cells can be found in Hall [74].

Chapter 2

A CLASSIFICATION SCHEME FOR ROBOTIC CELLS AND NOTATION

In this chapter, we discuss a classification scheme for sequencing and scheduling problems in robotic cells and provide notation. As in the classification scheme for classical scheduling problems (Graham et al. [69]), we distinguish problems based on three characteristics: machine environment (α), processing characteristics (β), and objective function (γ). A problem is then represented by the form $\alpha|\beta|\gamma$. Following the discussion of these characteristics, we detail the classification in Section 2.4 and provide a pictorial representation in Figure 2.2. Finally, we discuss relevant cell data whose values influence a cell's performance and define some basic notation for subsequent use.

2.1 Machine Environment

We start by describing characteristics that are represented in the first field of the classification scheme.

2.1.1 Number of Machines

If each processing stage has only one machine, the robotic cell is called a *simple robotic cell* or a *robotic flowshop*. Such a cell contrasts with a *robotic cell with parallel machines*, in which at least one processing stage has two or more identical machines. Cells with parallel machines are discussed in Chapter 5.

A typical simple robotic cell contains m processing machines: M_1, M_2, $\ldots$, M_m. Let $M = \{1, 2, \ldots, m\}$ be the set of indices of these machines.

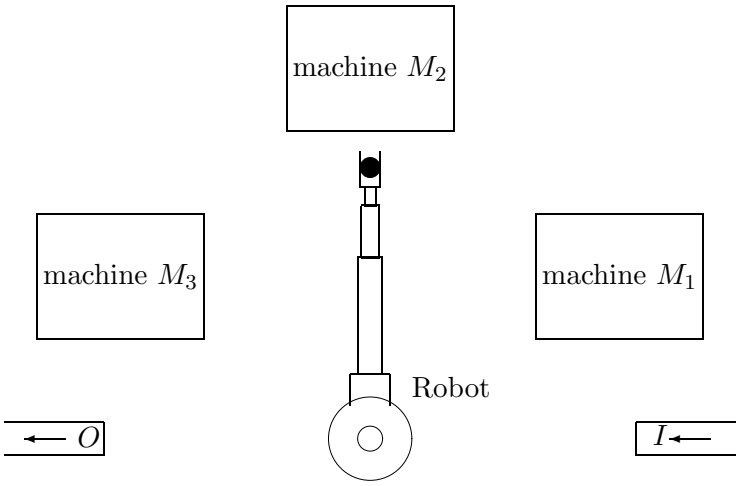


Figure 2.1. A Three-Machine Simple Robotic Cell.

The robot obtains a part from the input device (I, also denoted as M_0), carries the part to the first machine (M_1), and loads the part. After M_1 completes its processing on the part, the robot unloads the part from M_1 and transports it to M_2, on which it loads the part. This pattern continues for machines $M_3, M_4, \ldots, M_m$. After the last machine M_m has completed its processing on the part, the robot unloads the part and carries it to the output device (O, also denoted as M_{m+1}). In some implementations, the input device and the output device are at the same location, and this unit is called a *load lock*. A three-machine simple robotic cell is depicted in Figure 2.1.

This description should not be misconstrued as implying that the robot remains with each part throughout its processing by each machine. Often, after loading a part onto a machine, the robot moves to another machine or to the input device to collect another part to transport to its next destination. Determining which sequence of such moves maximizes the throughput of the cell has been the focus of the majority of research on robotic cell sequencing and scheduling.

2.1.2 Number of Robots

Manufacturers employ additional robots in a cell in order to increase throughput by increasing the material handling capacity. Cells with one (resp., more than one) robot are called *single-robot* (resp., *multiple-robot*)

cells. Most studies in the literature analyze single-robot cells. Multiple-robot cells are discussed in Chapter 8.

2.1.3 Types of Robots

A single-gripper robot can hold only one part at a time. In contrast, a dual-gripper robot can hold two parts simultaneously. In a typical use of this capability, the robot holds one part while the other gripper is empty; the empty gripper unloads a machine, the robot repositions the second gripper, and it loads that machine. Dual-gripper robots are discussed in Chapter 4.

In a single-gripper simple robotic cell, the robot cannot unload a part from machine $M_i, i = 0, ..., m-1$, unless the next machine M_{i+1} is empty. This condition is commonly referred to as a *blocking* condition.

2.1.4 Cell Layout

The layout refers to the arrangement of machines within the cell. Most robotic cell models assume one of two layouts: *linear* or *circular.* A semicircular arrangement of machines has also been referred to in the literature. However, all our results for a linear layout (Figure 1.3) remain valid for a semicircular layout (Figure 1.1). Unless specified otherwise, we assume a linear/semicircular layout. Cells employing a circular layout (Figure 1.5) are discussed in Chapters 4 and 7.

2.2 Processing Characteristics

Four different processing characteristics are specified in the second field. We describe three in this section. The fourth, called the production strategy, is detailed in Section 2.4.

2.2.1 Pickup Criterion

Most of the discussion in this book concerns robotic cells with no buffers for intermediate storage. For such cells, all parts must be either in the input device, on one of the machines, in the output device, or with the robot.

Robotic cells can be partitioned into three types − *free pickup, no-wait,* and *interval* − based on the pickup criterion. For all three types, a part that has completed processing on M_i cannot be loaded onto M_{i+1}

for its next processing unless M_{i+1} is unoccupied, $i = 0, \ldots, m$. In free-pickup cells, this is the only pickup restriction; there is no limit on the amount of time a part that has completed processing on a machine can remain on that machine.

For the more restrictive no-wait cells, a part must be removed from machine M_i, $i \in M$, and transferred to machine M_{i+1} as soon as M_i completes processing that part. Such conditions are commonly seen in steel manufacturing or plastic molding, where the raw material must maintain a certain temperature, or in food canning to ensure freshness (Hall and Sriskandarajah [77]). Results for no-wait cells are discussed in Chapter 9.

In interval robotic cells, each stage has a specific interval of time – a processing time window – for which a part can be processed at that stage. Thus, if $[a_i, b_i]$ is the processing time window at stage $i, i \in M$, then a part must be processed for a_i time units on stage i, and must be transferred to stage $(i+1)$ within $(b_i - a_i)$ time units after its completion of processing on stage i. This is applicable, for example, for the hoist scheduling problem on an electroplating line (Che et al. [30], Chen et al. [33], Lei and Wang [108]): printed circuit boards are placed in a series of tanks with different solvents. Each tank has a specific interval of time for which a card can remain immersed. Interval cells are discussed in Chapter 9.

Unless specified otherwise, the cells we discuss in the chapters that follow have the free-pickup criterion.

2.2.2 Travel-Time Metric

The robot's travel time between machines greatly influences a cell's performance. One common model often applies when the machines are arranged in numeric order in a line (Figure 1.3) or semicircle (Figure 1.1). The robot's travel time between adjacent machines M_{i-1} and M_i, denoted $d(M_{i-1}, M_i)$, equals δ, for $i = 1, \ldots, m+1$, and is *additive*. That is, the travel time between any two machines $M_i, M_j, 0 \leq i, j \leq m+1$ is $d(M_i, M_j) = |i - j|\delta$. This scheme is easily generalized to the case of unequal travel times between adjacent machines (Brauner and Finke [20]): $d(M_{i-1}, M_i) = \delta_i$, $i = 1, \ldots, m+1$, and $d(M_i, M_j) = \sum_{k=i+1}^{j} \delta_k$, for $i < j$. If $d(M_{i-1}, M_i) = \delta$, $i = 1, \ldots, m+1$, then we call the travel-

time metric *regular additive*. If $d(M_{i-1}, M_i) = \delta_i$, $i = 1, \ldots, m+1$, then the cell has *general additive* travel times.

There are also additive travel-time cells in which the machines are arranged in a circle so that I and O are adjacent or in the same location (Drobouchevitch et al. [50], Geismar et al. [61], Sethi et al. [143], Sriskandarajah et al. [146]). In these cells, the robot may travel in either direction to move from one machine to another; e.g., to move from M_1 to M_{m-1}, it may be faster to go via I, O, and M_m, than to go via $M_2, M_3, \ldots, M_{m-2}$. For circular cells with regular additive travel times, $d(M_i, M_j) = \min\{|i-j|\delta, (m+2-|i-j|)\delta\}$. For general additive travel-time cells, $d(M_i, M_j) = \min\{\sum_{k=i+1}^{j} \delta_k, \sum_{k=1}^{i} \delta_k + \delta_{0,m+1} + \sum_{k=j+1}^{m+1} \delta_k\}$ for $i < j$. Most studies assume that the travel times are symmetric, i.e., $d(M_i, M_j) = d(M_j, M_i), 0 \leq i, j \leq m+1$, and that the travel time between any two machines does not depend on whether or not the robot is carrying a part.

To make this model better represent reality, it can be enhanced to account for the robot's acceleration and deceleration (Logendran and Sriskandarajah [116]). The travel times between adjacent machines do not change. However, the travel time between nonadjacent machines is reduced. For each intervening machine, the robot is assumed to save η units of time. Therefore, for $0 \leq i, j \leq m+1$, if $d(M_{i-1}, M_i) = \delta_i$, then

$$d(M_i, M_j) = \sum_{k=\min(i,j)+1}^{\max(i,j)} \delta_k - (|i-j|-1)\eta.$$

We use this model in our discussions in Chapter 6.

For certain cells, additive travel times are not appropriate. Dawande et al. [47] discuss a type of cell for which the robot travel time between *any* pair of machines is a constant δ, i.e., $d(M_i, M_j) = \delta, 0 \leq i, j \leq m+1$, $i \neq j$. This arises because these cells are compact and the robots move with varying acceleration and deceleration between pairs of machines.

The most general model, one that can represent all the travel-time metrics typically encountered in practice, assigns a value δ_{ij} for the robot travel time between machines M_i and $M_j, 0 \leq i, j \leq m+1$. These travel times are, in general, neither additive nor constant. Brauner et al. [24] address this problem by making three assumptions that conform to basic properties of Euclidean space:

1. The travel time from a machine to itself is zero, that is, $\delta_{ii} = 0, \forall i$.
2. The travel times satisfy the triangle inequality, that is, $\delta_{ij} + \delta_{jk} \geq \delta_{ik}, \forall i, j, k$.
3. The travel times are symmetric, that is, $\delta_{ij} = \delta_{ji}, \forall i, j$.

A robotic cell that satisfies Assumptions 1 and 2 is called a *Euclidean robotic cell*, and one that satisfies Assumptions 1, 2, and 3 is called a *Euclidean symmetric robotic cell*. As we shall discuss in Chapter 3, the robot move sequencing problem for either case is strongly NP-hard (Brauner et al. [24]). This is also why most studies approximate reality with additive or constant travel-time models, depending on which of the two is a better fit.

To summarize, three different robot travel-time metrics have been addressed in the literature: additive, constant, and Euclidean. Most studies assume one of these. Therefore, many results in the field have been proven only for one travel-time metric rather than for all three.

2.2.3 Number of Part-Types

A cell producing identical parts is referred to as a *single-part-type cell*. In contrast, a *multiple-part-type cell* processes lots that contain different types of parts. Generally, these different part types require different processing times on a given machine. Multiple-part-type cells are discussed in Chapters 6 and 7. Throughout the rest of the book, unless specified otherwise, the cell under consideration processes identical parts.

2.3 Objective Function

From an optimization aspect, the objective that is predominantly addressed in the literature is that of maximizing the *throughput* − the long-term average number of completed parts placed into the output buffer per unit time. This will be our objective throughout the book. A precise definition of throughput is provided in Chapter 3.

2.4 An $\alpha|\beta|\gamma$ Classification for Robotic Cells

Figure 2.2 is a pictorial representation of the classification discussed in the preceding text. A problem is represented using the form $\alpha|\beta|\gamma$, where

(a) $\alpha = RF^{g,l}_{m,r,\bar{b}}(m_1, ..., m_m)$. Here, RF stands for "Robotic Flowshop," m is the number of processing stages, and the vector $(m_1, m_2, ..., m_m)$ indicates the number of identical machines at each stage. When this vector is not specified, $m_i = 1, i = 1, ..., m$, and the cell is a simple cell. The second subscript r denotes the number of robots; when not specified, $r = 1$. For cells with output buffers at the various stages of the cell, the vector $\bar{b} = (b_1, ..., b_m)$ denotes the sizes of the buffers. At stage i, the size of the output buffer is denoted by $b_i, i = 1, ..., m$; this notation is omitted for cells without buffers. The first superscript g denotes the type of robot used. For example, $g = 1$ (resp., $g = 2$) denotes a single-gripper (resp., dual-gripper) cell. If g is not specified, then $g = 1$. The second superscript l indicates the layout of the cell; a linear/semicircular (resp., circular) layout is indicated by $\sqcup$ (resp., $\circ$). Most of our discussion is for linear or semicircular layouts; unless specified otherwise, such a layout is assumed, and the notation is omitted.

(b) $\beta =$(*pickup, travel-metric, part-type, prod-strategy)*, where

- *pickup* $\in$ {*free, no-wait, interval*} specifies the pickup criterion.
- *travel-metric* $\in$ {*A, C, E*} specifies the travel-time metric, where *A*, *C*, and *E* denote the additive, the constant, and the Euclidean travel-time metric, respectively.
- If *part-type* is not specified, the cell produces a single part-type. Otherwise, *part-type* = *MP* denotes a cell producing multiple part-types.
- *prod-strategy* $\in$ {*cyclic-k, LCM, all, CRM*} denotes the specific production strategy employed. The detailed descriptions of these strategies appear in later chapters, so we limit our description here and refer the reader to the corresponding chapter.

  (i) In a cell producing either a single part-type or multiple part-types, *cyclic-k* refers to a *cyclic production* strategy wherein exactly k units are produced per cycle (Chapter 3). When the integer k is not specified, the production strategy includes all k-unit cycles, $k \geq 1$. *LCM cycles* form a subclass of cyclic solutions, and are discussed in Chapters 5 and 8.

(ii) In a cell producing either a single part-type or multiple part-types, *all* refers to a production environment where all production strategies (i.e., cyclic as well as noncyclic) are considered (Chapters 3 and 10).

(iii) In robotic cells producing multiple part-types (Chapters 6 and 7), *CRM* refers to the *concatenated robot-move sequence* strategy.

(c) $\gamma = \mu$ denotes the objective function of maximizing the throughput. Although this is the only objective function addressed in our discussion, we use a separate field to allow for future work involving different objective functions.

We now illustrate our classification with a few examples.

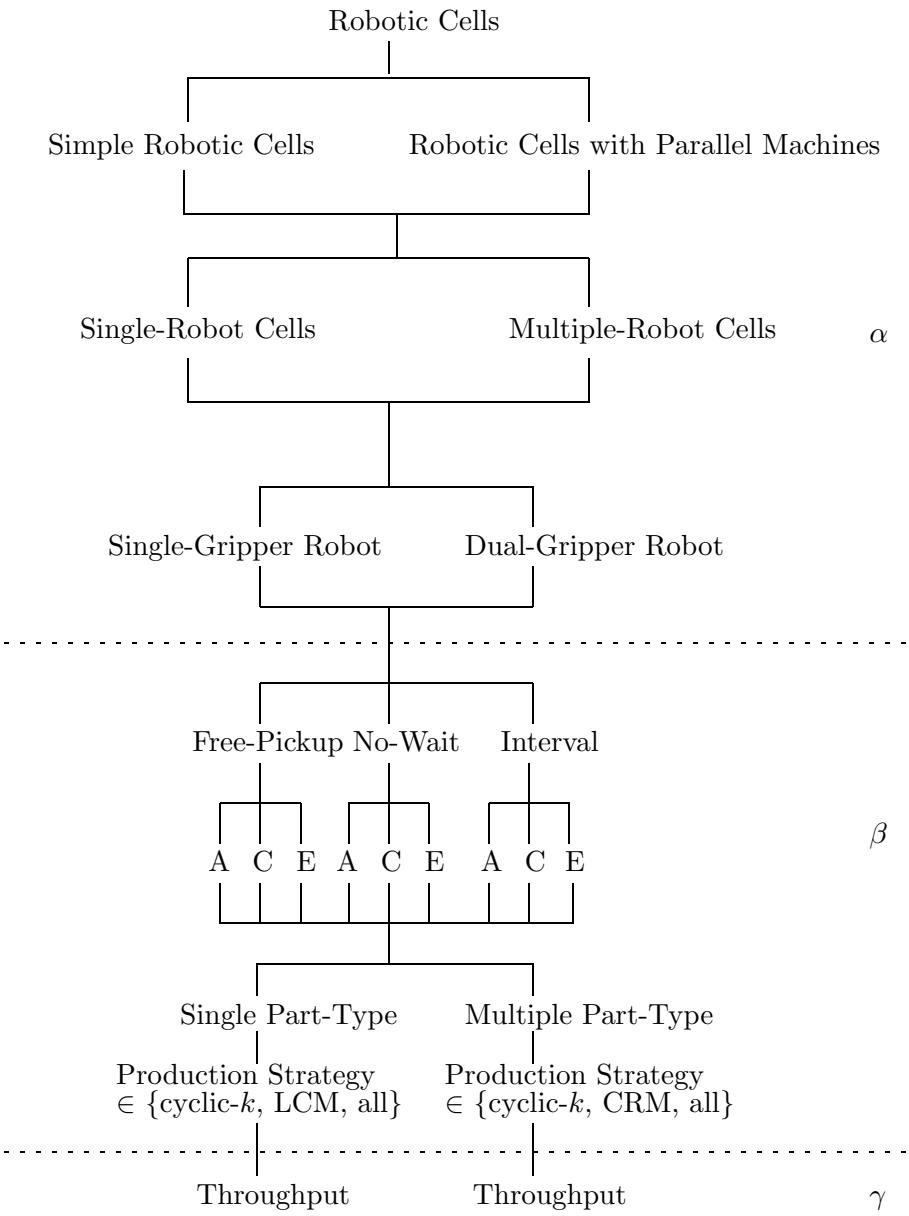
1. $RF_4|(\textit{free,A,cyclic-1})|\mu$ represents a four-machine simple robotic cell with one single-gripper robot, a free-pickup criterion, and additive travel-time metric. It produces a single part-type and operates a cyclic production strategy wherein one unit is produced per cycle. The objective function is that of maximizing the throughput.

2. $RF_5(1,4,2,3,2)|(\textit{no-wait,E,cyclic-2})|\mu$ refers to the problem of maximizing throughput for a five-stage robotic cell with parallel machines that has one, four, two, three, and two machines, respectively, in stages 1, 2, 3, 4, and 5. The cell produces a single part-type, has one single-gripper robot, employs a no-wait pickup criterion and a Euclidean travel-time metric, and produces two units per cycle.

3. $RF^2_{m,3}|(\textit{interval,C,MP,CRM})|\mu$ considers the problem of throughput maximization in an m-machine simple robotic cell with three dual-gripper robots, an interval pickup criterion, constant travel-time metric, and multiple-part-type production using a CRM production strategy.

4. $RF^{2,\circ}_{m,\bar{1}}|(\textit{free,A,cyclic-k})|\mu$ is the problem of maximizing the throughput over all cyclic schedules in an m-machine dual-gripper cell with an output buffer of size one at each machine. The travel-time metric is additive, and the layout of the cell is circular.

In the chapters that follow, we use this classification to specify the problem under consideration.



A, C, E denote Additive, Constant, Euclidean Travel Time, respectively

Figure 2.2. A Classification of Robotic Cells

2.5 Cell Data

In addition to the robot's travel-time metric, the processing times at the various stages and the times required for loading and unloading a machine influence the cell's throughput. We now discuss these characteristics and the notation for representing the actions of the robot and the states of the cell. First, we list the basic assumptions throughout most studies:

- All data and events are deterministic.
- All processing is nonpreemptive.
- Parts to be processed are always available at the cell's input device.
- There is always space for completed parts at the output device.
- All data are rational.

2.5.1 Processing Times

Since each of the m stages performs a different function, each, in general, has a different processing time for a given part. For cells with free pickup or no-wait pickup, the processing time of a machine in stage j is denoted by $p_j, j \in M$. If a cell processes k different types of parts, the processing time of part i at stage j is denoted by $p_{ij}, i = 1, \ldots, k; j \in M$. In interval robotic cells, the processing time of machine M_j is specified by a lower bound l_j and an upper bound $u_j \geq l_j$. For example, the time that a printed circuit board spends in tank j must be in the interval $[l_j, u_j]$. If multiple part-types are processed in an interval robotic cell, the processing interval for part-type i is denoted by $[l_{ij}, u_{ij}]$.

2.5.2 Loading and Unloading Times

Another factor that influences the processing duration for a part is the time required for loading and unloading at each machine. For uniformity, picking a part from I is referred to as unloading I, and dropping a part at O is referred to as loading O. Typically, models assume that the loading and unloading times are equal (ϵ) for all machines. This will be our assumption as well for most of the discussion. More sophisticated models have different values for loading and unloading at each machine:

the loading (resp., unloading) time for M_i is ϵ_{2i} (resp., ϵ_{2i+1}), $i = 1, ..., m$; the unloading (resp., loading) time at I (resp., O) is ϵ_1 (resp., $\epsilon_{2(m+1)}$). We use this more general notation in Chapter 6.

2.5.3 Notations for Cell States and Robot Actions

By the state of the robotic cell at any given instant of time, we mean a sufficient description of the cell required for the purpose of our analysis. To keep the notation simple, our discussion in this section is limited to simple robotic cells with the free-pickup criterion; appropriate enhancements can be made for other classes of cells.

Ideally, a precise mathematical description of the state of the cell would include the following.

- The occupancy state of each machine. That is, whether a machine contains a part or it is empty.
- If a machine contains a part, then the time remaining on its current processing.
- The location of the robot.
- The occupancy state of the robot, that is, whether the robot arm has a part or not.

Before we formalize the state space, note that since we are interested in maximizing the throughput of the cell, it is not necessary to consider "wasteful" robot actions such as unnecessary waiting at a location or moving to a location without performing at least one of the loading or unloading operations. Also, since this is a deterministic problem, it is sufficient to define decisions regarding the robot's moves only at those epochs when the robot has just finished loading or unloading a part at a machine. It follows that it is sufficient to consider the state when the robot's position is at these epochs.[1]

Our focus in this book is on a steady-state analysis of a certain class of solutions referred to as cyclic solutions (discussed in Chapter 3). Typ-

[1]In the stochastic setting, say when the processing times are random variables, a throughput maximizing operation may require the robot arm to change its traversal path while the robot is in transition, at a time when some new information becomes available. To allow for this, a continuous state space and continuous decision making over time are required.

ically, this analysis does not require a detailed state description since the definition of cyclic solutions involves a requirement that completes the missing information. In view of this, all that is needed is a specification of each machine in terms of whether it is occupied or not. Such a simplified description can be presented as an $(m+1)$-dimensional vector $(e_1, \ldots, e_{m+1})$ (Sethi et al. [142]). Each of the first m dimensions corresponds to a machine: $e_i = \emptyset$ if M_i is unoccupied; $e_i = \Omega$ if M_i is occupied, $i = 1, \ldots, m$. The last dimension represents the robot; $e_{m+1} = M_i^-$ indicates that the robot has just completed loading a part onto $M_i, i = 1, \ldots, m+1$, and $e_{m+1} = M_i^+$ indicates that the robot has just completed unloading a part from $M_i, i = 0, \ldots, m$.

EXAMPLE 2.1 For $m = 4$, consider the state $(\emptyset, \Omega, \emptyset, \Omega, M_2^-)$: M_1 and M_3 are unoccupied, M_2 and M_4 are occupied, and the robot has just completed loading M_2. Suppose that the robot's next actions were to travel to I, unload a part from I, travel to M_1, and load that part onto M_1. The states corresponding to these actions are $(\emptyset, \Omega, \emptyset, \Omega, M_0^+)$ and $(\Omega, \Omega, \emptyset, \Omega, M_1^-)$. Note that listing the state $(\emptyset, \Omega, \emptyset, \Omega, M_0^+)$ is superfluous. To transition from $(\emptyset, \Omega, \emptyset, \Omega, M_2^-)$ into $(\Omega, \Omega, \emptyset, \Omega, M_1^-)$, the robot must have first traveled to I.

In general, a series of robot actions can be completely represented by a string of M_i^- symbols. For example, $M_2^- M_4^- M_5^-$ means that the robot unloads a part from M_1, travels to M_2, and loads the part onto M_2. The robot next travels to M_3, waits for M_3 to finish processing (if required), unloads a part from M_3, travels to M_4, and loads the part onto M_4. The robot waits at M_4 while the part is being processed. The robot then unloads the part from M_4, carries it to M_5, and loads M_5.

A different notation has largely supplanted the M_i^- notation in the literature. This more popular notation is based on the concept of an *activity*. Activity A_i consists of the following sequence of actions:

- The robot unloads a part from M_i.
- The robot travels from M_i to M_{i+1}.
- The robot loads this part onto M_{i+1}.

The sequence of actions discussed above ($M_2^- M_4^- M_5^-$) would be represented by $A_1 A_3 A_4$. Since a part must be processed on all m machines and then placed into the output buffer, one instance of each of the $m+1$ activities $A_0, A_1, \ldots, A_m$, is required to produce a part.

It is easy to use the activity-based notation to represent the cell's current status. Let $e_{m+1} = A_i$ indicate that the robot has just completed activity A_i; $e_i, i = 1, 2, \ldots, m$, will have the same meaning as before.

EXAMPLE 2.2 For $m = 4$, an example state is $(\Omega, \emptyset, \emptyset, \Omega, A_3)$: M_2 and M_3 are unoccupied, M_1 and M_4 are occupied, and the robot has just completed loading M_4. From this point, let us now consider what happens if the robot executes activity sequence $A_1 A_2 A_4$: the robot moves to M_1, waits (if required) for M_1 to finish processing, unloads a part from M_1, travels to M_2, and loads the part onto M_2. At this instant, the state of the cell is $(\emptyset, \Omega, \emptyset, \Omega, A_1)$. The robot waits at M_2 for the entirety of the part's processing. The robot then unloads the part from M_2, carries it to M_3, and loads the part onto M_3. The cell's state is now $(\emptyset, \emptyset, \Omega, \Omega, A_2)$. The robot next travels to M_4, waits (if required) for M_4 to finish processing, unloads a part from M_4, travels to the output buffer, and loads the part onto the output buffer, so the cell's state is $(\emptyset, \emptyset, \Omega, \emptyset, A_4)$.

For most of the discussion in this book, we will represent robot actions by using the activity notation: $A_i, i = 0, 1, ..., m$. The discussion for robotic cells producing multiple part-types, however, is easier with the M_i^- notation; we will use it in Chapters 4 and 6. The M_i^- notation is also convenient for describing moves in a dual-gripper robotic cell (Chapter 4).

The simplified state description above omits information representing the extent of the processing completed on the parts on the various machines. A more precise representation of a state is an $(m+1)$-tuple $\Gamma = (s_1, \ldots, s_{m+1})$, where $s_i \in \{-1, r_i\}, i \in M$. If $s_i = -1$, machine M_i has no part on it; otherwise r_i is the time remaining in the processing of the current part on M_i. As before, $s_{m+1} \in \{A_i, i = 0, ..., m\}$ denotes that the robot has just completed activity A_i (i.e., loaded a part onto machine M_{i+1}).

EXAMPLE 2.3 For $m = 4$, the state vector $\Gamma = (5, 0, -1, p_4, A_3)$ indicates that the part on M_1 has five time units of processing remaining, M_2 has completed processing a part and that part still resides on M_2, and M_3 is empty. The robot has unloaded a part from M_3, carried it to M_4, and just completed loading it onto M_4.

There is another important observation to be made here. Note that even with integer data, the remaining processing times are in general real numbers. However, since we need to consider the system state only at the epochs mentioned above, the state description will be integral provided the initial state of the system is restricted to be in integer terms. This restriction can be imposed without loss of generality since some initial adjustments can be made at the beginning to bring the state to integral terms, and the time taken to make these adjustments is of no consequence in the context of the long-term average throughput criterion. Thus, in any state description $\Gamma = (s_1, \ldots, s_{m+1})$, $s_i \in \{-1, r_i\}$ with $r_i \in \{k \in \mathbb{Z} : 0 \leq k \leq p_i\}$, $i \in M$. We thus have a *finite-state* dynamic system.

Chapter 3

CYCLIC PRODUCTION

Cyclic production in a robotic cell refers to the production of finished parts by repeating a fixed sequence of robot moves. More precisely, for an integer $k \geq 1$, a typical operation of the cell consists of a sequence of robot moves in which exactly k parts are taken from the input device M_0, exactly k parts are dropped at the output device M_{m+1}, and the cell returns to its initial state, i.e., the state at the beginning of the sequence. A particular sequence of robot moves is chosen and repeated until the required production is complete. In practice, such cyclic schedules are easy to implement and control, and are the primary way of specifying the operation of a robotic cell.

In this chapter, we consider cyclic production of identical parts. We start by proving the sufficiency of considering cyclic schedules. Section 3.2 illustrates the computation of cycle time. In Section 3.3, we examine the optimality of 1-unit cycles. In Section 3.4, we briefly visit the issue of computing the makespan of a lot. Section 3.5 is devoted to obtaining upper bounds on the ratio of the throughputs of an optimal cyclic solution and an optimal 1-unit cycle.

3.1 Operating Policies and Dominance of Cyclic Solutions

Under the assumption of rational (or, equivalently, integer) data, it is easy to establish the sufficiency of considering the class of cyclic schedules to maximize throughput over all schedules (Dawande et al. [45]). To do so, we analyze the operations of a robotic cell as a sequence of states,

rather than as a sequence of activities. For simplicity, we limit our discussion here to simple robotic cells with free pickup; extensions to other classes of cells are straightforward.

Recall our discussion of Section 2.5.3; in particular, the (i) sufficiency of considering non-wasteful robot actions and (ii) sufficiency of considering the state of the cell only at the epochs defined by the following state description: a state of the cell is an $(m+1)$-tuple $\Gamma = (s_1, \ldots, s_{m+1})$, where $s_i \in \{-1, r_i\}, i \in M; r_i \in \{k \in \mathbb{Z} : 0 \leq k \leq p_i\}$. If $s_i = -1$, machine M_i has no part on it; otherwise $s_i = r_i$ is the time remaining in the processing of the current part on M_i. Finally, $s_{m+1} \in \{A_i, i = 0, ..., m\}$ denotes that the robot has just completed activity A_i (i.e., loaded a part onto machine M_{i+1}). Let $\mathcal{F}$ denote the set of all feasible states.

DEFINITION 3.1 An *operating sequence* for the cell is an infinite sequence of successive states resulting from feasible operations of the cell starting from an initial state.

It is important to note that not every infinite sequence of states is feasible. For example, the state $\Gamma_1 = (5, 0, -1, p_4, A_3)$ followed immediately by $\Gamma_2 = (5, 0, -1, 0, A_4)$ results in an infeasible sequence if $p_4 + \delta_{45} + 2\epsilon > 0$. The reason is as follows. Since Γ_2 is the next state of the cell after state Γ_1, after the robot loads a part onto machine M_4 (state Γ_1), it waits at that machine for the entire duration while M_4 is processing the part. The robot then unloads the part from M_4 and loads it onto M_5. However, since machine M_1 is busy processing its part during this time, at the instant the robot finishes loading machine M_5 (state Γ_2), the processing time remaining on M_1 is $\max\{0, 5 - p_4 - \delta_{45} - 2\epsilon\} < 5$.

DEFINITION 3.2 A *policy* for the cell is a function d: $\mathcal{F} \rightarrow \mathcal{F}$ such that there exists a state $\Gamma \in \mathcal{F}$ for which the infinite sequence $T(d, \Gamma) \equiv \{\Gamma, d(\Gamma), d^2(\Gamma), ..., d^n(\Gamma), ...\}$ is an operating sequence.

Consider an optimal operating sequence, say Σ, and suppose that there exists no policy that can generate it. Then, for some state Γ, the action taken by Σ is different at two (or more) instances when the cell is in state Γ. Without loss of generality, we can assume that state Γ occurs in Σ infinitely often, for if the number of occurrences of a state is finite, the segment of Σ up to the last instance of that state can be deleted without affecting its long-term throughput. Let *segment-*

throughput refer to the average number of finished parts produced per unit of time for the segments of Σ between two successive occurrences of Γ. If all of the segment-throughputs are equal, we can replace each of these segments by any one segment and maintain the throughput of Σ. Otherwise, replacing a segment having a smaller value of segment-throughput with one having a larger value contradicts the optimality of Σ. We thus have the following result.

LEMMA 3.1 *There exists a throughput maximizing operating sequence that can be generated by a policy.*

Given that the cell is currently in state $\Gamma \in \mathcal{F}$, the functional image $d(\Gamma)$ of a policy d completely specifies the transition to the next state, and thus completely defines the robot's action. Together, a policy d and an initial state $\Gamma_0 \in \mathcal{F}$ generate a unique operating sequence $\{\Gamma_0, d(\Gamma_0), d^2(\Gamma_0), ..., d^n(\Gamma_0), ...\}$. We would like to emphasize that an initial state is required to specify an operating sequence generated by a policy. To illustrate, suppose $\mathcal{F} = \{\Gamma_1, \Gamma_2, ..., \Gamma_6\}$ and d is defined as follows: $d(\Gamma_i) = \Gamma_{i+1}, i = 1, 2, 4, 5; d(\Gamma_3) = \Gamma_1, d(\Gamma_6) = \Gamma_4$. If the initial state is Γ_1, we obtain the sequence $\{\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_1, \Gamma_2, \Gamma_3, ...\}$. If the initial state is Γ_4, we obtain $\{\Gamma_4, \Gamma_5, \Gamma_6, \Gamma_4, \Gamma_5, \Gamma_6, ...\}$.

Let $\mu(d, \Gamma)$ be the throughput of the operating sequence $T(d, \Gamma)$. The maximum throughput, $\mu(d)$, obtainable from a policy d is then

$$\max_{\Gamma \in \mathcal{F}} \{\mu(d, \Gamma)\colon\ T(d, \Gamma) \text{ is an operating sequence}\}.$$

Note that the maximum exists since $|\mathcal{F}|$ is finite. The maximum throughput of the cell is obtained by maximizing $\mu(d)$ over all policies d. Since a policy is a function with domain and range on the finite set $\mathcal{F}$, the total number of distinct policies is at most $|\mathcal{F}|^{|\mathcal{F}|}$. Moreover, since an operating sequence is completely specified by a policy and an initial state, the total number of operating sequences is at most $|\mathcal{F}|^{(|\mathcal{F}|+1)}$. The finiteness of $\mathcal{F}$ implies that the infinite sequence of states resulting from *any* policy is a repeating sequence. Consequently, the sequence of robot actions is a repeating sequence. Every policy repeats a minimal sequence of robot moves. The minimal sequence is a state-preserving sequence: the state of the cell at the beginning is identical to the state of the cell at the end of the sequence. The discussion above and Lemma 3.1 yield the following result.

THEOREM 3.1 *There exists a cyclic sequence of robot moves that maximizes long-term throughput of the robotic cell.*

It is therefore sufficient to optimize over the class of cyclic sequences. This result provides a sound justification for the widely used industry practice of specifying the operation of a robotic cell via cyclic sequences. Using the notation defined in the previous chapter, cyclic production can be represented as a repeatable sequence of activities. For example, (A_0, A_2, A_4, A_3, A_1) is a sequence of activities that produces a part in a four-machine cell. Such a sequence can be repeated in a cyclic fashion, with each iteration producing a single part. To formalize, we define the following terms:

DEFINITION 3.3 A *k-unit activity sequence* is a sequence of robot moves that loads and unloads each machine exactly k times.

To be feasible, an activity sequence must satisfy two criteria:

- The robot cannot be instructed to load an occupied machine
- The robot cannot be instructed to unload an unoccupied machine.

These concepts are operationalized as follows: *During cyclic operations, for $i = 1, \ldots, m-1$, between any two occurrences of A_i there must be exactly one A_{i-1} and exactly one A_{i+1}.* This condition implies that between any two instances of A_0 there is exactly one A_1, and between any two instances of A_m there is exactly one A_{m-1}. For instance, in a cell with $m = 3$, the 2-unit activity sequence $(A_0, A_1, A_3, A_1, A_2, A_0, A_3, A_2)$ is infeasible because the second occurrence of A_1 attempts to unload machine M_1 when it is empty. Note that all 1-unit activity sequences are feasible.

DEFINITION 3.4 A *k-unit cycle* is the performance of a feasible k-unit activity sequence in a way that leaves the cell in exactly the same state as its state at the beginning of those moves.

For every feasible k-unit activity sequence, $k \geq 1$, there is at least one initial state for which it is a k-unit cycle, i.e., if the k-unit activity sequence begins with this state, it leaves the cell in exactly the same state after its execution [146]. Since a k-unit cycle preserves the state of the cell, repeating it indefinitely yields a *k-unit cyclic solution.* A cyclic

solution is also known as a *steady-state* solution. We provide a more rigorous definition of steady state below.

A k-unit activity sequence has $k(m+1)$ activities; each of the $m+1$ activities $A_i, i = 0, 1, \ldots, m$, is performed exactly k times and in an order that satisfies the feasibility constraints. A k-unit cycle constructed from a k-unit activity sequence $(A_0, A_{i_1}, A_{i_2}, \ldots, A_{i_{k(m+1)-1}})$ will be referred to as the k-unit cycle or, simply, cycle $(A_0, A_{i_1}, A_{i_2}, \ldots, A_{i_{k(m+1)-1}})$. Since a k-unit cyclic solution is completely characterized by a k-unit cycle, we will use the two terms interchangeably when no confusion arises in doing so.

Define the function $F(A_i, t)$ to represent the time of completion of the t^{th} execution of activity A_i [40]. Given a feasible infinite sequence of activities and a compatible initial state, we can define the *long-run average throughput* or, simply, *throughput* to be

$$\mu = \lim_{t \to \infty} \frac{t}{F(A_m, t)}.$$

Intuitively, this quantity represents the long-term average number of completed parts placed into the output buffer per unit time. Obtaining a feasible infinite sequence of activities that maximizes throughput is a fundamental problem of robotic cell scheduling. Such a sequence of robotic moves is called *optimal*. Most studies focus on infinite sequences of activities in which a fixed sequence of $m+1$, or some integral multiple of $m+1$, activities is repeated cyclically.

Definition 3.5 [40] A robotic cell repeatedly executing a k-unit cycle π of robot moves is operating in *steady state* if there exists a constant $T(\pi)$ and a constant N such that for every A_i, $i = 0, \ldots, m$, and for every $t \in \mathbb{Z}^+$ such that $t > N$, $F(A_i, t+k) - F(A_i, t) = T(\pi)$. $T(\pi)$ is called the *cycle time* of π.

For additive travel-time cells, we denote the cycle time by $T_a(\pi)$. The corresponding notation for constant and Euclidean travel-time cells will be $T_c(\pi)$ and $T_e(\pi)$, respectively.

The *per unit cycle time* of a k-unit cycle π is $T(\pi)/k$. This is the reciprocal of the throughput and is typically easier to calculate directly. Therefore, minimizing the per unit cycle time is equivalent to maximizing the throughput.

An assumption in most studies is that the sequence of robot moves is *active*. A sequence is called active if the robot always executes the next operation, whatever that may be, as soon as possible. For active sequences, all execution times for the robot's actions are uniquely determined once the sequence of activities is given. The robot's only possible waiting period can occur at a machine at which the robot has arrived to unload, but the machine has not completed processing its current part. In the class of optimal robot move sequences, there is at least one active sequence [153].

Brauner and Finke [22] show that repeating a k-unit activity sequence will enable the robotic cell to reach a steady state (or cyclic solution) in finite time. Therefore, since we are maximizing the long-run average throughput, i.e., assuming that the cell operates in steady state for an infinite time, there is no impact from the initial transient phase [45, 76]. Hence, there is no loss of generality by studying only the steady-state behavior. Nevertheless, there may be some practical reason to find the time required to reach steady state. This is discussed in Chapters 6 and 7.

3.2 Cycle Times

In this section, we discuss the robot's waiting time at a machine and methods for computing the cycle time of a given cycle. We also establish lower bounds for the cycle time. For simplicity of exposition, the discussion is limited to 1-unit cycles in simple robotic cells.

3.2.1 Waiting Times

The robot waits at a machine M_i if its next sequenced action is to unload M_i, but M_i has not yet completed processing its current part. The length of the robot's *waiting time*, denoted w_i, is M_i's processing time p_i minus the time that elapses between when M_i was loaded and when the robot returns to unload it. If this difference is negative, then the waiting time is zero.

The time that elapses between M_i's loading and the robot's return is determined by the intervening activities that are executed between the loading and the unloading of M_i. If there are no intervening activities, the robot loads M_i, waits at M_i for time p_i, then unloads M_i. Such a

sequence is represented by $A_{i-1}A_i$. In this case, M_i is said to have *full waiting* [47].

If there are intervening activities between the loading and the unloading of M_i, then M_i has *partial waiting* [47]. Consider the sequence $A_{i-1}A_jA_i$. The robot loads M_i, travels to M_j ($\delta_{i,j}$), waits for M_j to complete processing (w_j), unloads M_j (ϵ), carries that part to M_{j+1} ($\delta_{j,j+1}$), loads M_{j+1} (ϵ), then travels to M_i ($\delta_{j+1,i}$). The robot's waiting time at M_i is

$$w_i = \max\{0, p_i - \delta_{i,j} - w_j - \epsilon - \delta_{j,j+1} - \epsilon - \delta_{j+1,i}\}.$$

For a constant travel-time cell, this expression simplifies to $w_i = \max\{0, p_i - 3\delta - 2\epsilon - w_j\}$.

The expression for the robot's waiting time is often dependent on the waiting time at one or more machines. This recursion makes calculating the cycle time difficult. However, the condition that a cycle begins and ends in the same state allows us to uniquely compute the cycle time, as demonstrated in the next section.

3.2.2 Computation of Cycle Times

The cycle time is calculated by summing the robot's movement times, the loading and unloading times, and the robot's waiting times (full and partial). A straightforward approach for computing the cycle time of a given cycle requires solving a linear program. We illustrate this approach below for 1-unit cycles. Appendix A lists all 1-unit cycles for simple robotic cells with two, three, and four machines.

For each activity $A_i, i = 0, \ldots, m$, the robot unloads M_i, carries the part to M_{i+1}, and loads M_{i+1}. The total time for A_i is $\delta_{i,i+1} + 2\epsilon$. We must also account for the time between activities. If M_i has full waiting (A_{i-1} immediately precedes A_i), the robot spends exactly p_i time units between activities A_{i-1} and A_i waiting at M_i. If M_i has partial waiting (A_j immediately precedes $A_i, j \neq i-1$), then the robot moves from M_{j+1} to M_i ($\delta_{j+1,i}$) and waits for M_i to complete processing (w_i) before starting activity A_i.

For specificity, we now consider a constant travel-time cell with constant loading and unloading times. Let V_1 be the set of machines with full waiting, and V_2 be the set of those with partial waiting. The cycle

time for a 1-unit cycle is

$$T_c(\pi) = (m+1)(\delta + 2\epsilon) + \sum_{i \in V_1} p_i + \sum_{i \in V_2} (w_i + \delta) + \delta. \tag{3.1}$$

The extra δ accounts for the last movement of the cycle, which takes the robot to I to collect a new part.

For example, consider the cycle $\pi_3 = (A_0, A_1, A_3, A_2)$. $V_1 = \{1\}$, $V_2 = \{2, 3\}$, $m = 3$. The cycle time is

$$\begin{aligned}
T_c(\pi_3) &= 4(\delta + 2\epsilon) + p_1 + w_2 + w_3 + 3\delta \\
&= 7\delta + 8\epsilon + p_1 + w_2 + w_3, \qquad \text{where} \\
w_2 &= \max\{0, p_2 - 3\delta - 2\epsilon - w_3\}, \\
w_3 &= \max\{0, p_3 - 4\delta - 4\epsilon - p_1\}, \\
w_2 + w_3 &= \max\{0, p_2 - 3\delta - 2\epsilon, p_3 - 4\delta - 4\epsilon - p_1\}. \\
\text{Thus, } T_c(\pi_3) &= \max\{7\delta + 8\epsilon + p_1, 4\delta + 6\epsilon + p_1 + p_2, 3\delta + 4\epsilon + p_3\}.
\end{aligned}$$

Similarly, the cycle time for the cycle $\pi_6 = (A_0, A_3, A_2, A_1)$ is $T_c(\pi_6) = \max\{8\delta + 8\epsilon, p_1 + 3\delta + 4\epsilon, p_2 + 3\delta + 4\epsilon, p_3 + 3\delta + 4\epsilon\}$. Writing the equations for the waiting times requires that the cycle begin and end in the same state. In general, this method can be implemented as a linear program with km variables and km constraints, where k is the number of units produced in one cycle [61, 102]. Hence, it has time complexity $O((km)^3 L)$, where L is the size of the problem's binary encoding. For example, the waiting times required for computing $T_c(\pi_3)$ can be obtained by solving the following linear program.

$$\begin{aligned}
\text{Minimize} \quad & w_2 + w_3 \\
\text{subject to:} \quad & \\
w_2 + w_3 &\geq p_2 - 3\delta - 2\epsilon \\
w_3 &\geq p_3 - 4\delta - 4\epsilon - p_1 \\
w_2, w_3 &\geq 0
\end{aligned}$$

Another linear programming approach that directly deduces the exact time at which each machine is loaded and unloaded is described in [39].

From a point of view of computational complexity, the linear programming approach above is not the most efficient. There are more efficient graphical methods that find the cycle time without considering

robot waiting times. These methods analyze a cyclic graph resulting from the precedence relationships between the activities of the given cycle. We illustrate this approach on the 1-unit cycle $\pi = (A_0, A_2, A_1, A_3)$ in a three-machine simple robotic cell with free pickup and constant inter-machine travel time.

EXAMPLE 3.1 Given $\pi = (A_0, A_2, A_1, A_3)$, fix an arbitrary value T for its cycle time. A directed graph G (see Figure 3.1) is constructed as follows: for each machine $M_i, i = 1, 2, 3$, we have one vertex, v_i^l, signifying the beginning of each load operation and one vertex, v_i^u, signifying the beginning of each unload operation. In addition, we have two vertices, v_0^u and v_4^l, corresponding to the beginning of the unload and load operations on the input and output, respectively. The edge set consists of the following:

- Edges $(v_i^u, v_{i+1}^l), i = 0, 1, 2, 3$, corresponding to activities $A_i, i = 0, 1, 2, 3$. The weight of each of these edges is $\delta + \epsilon$, the time to execute each activity.
- Edges (v_0^l, v_2^u), (v_3^l, v_1^u), (v_2^l, v_3^u) corresponding to the cycle π. The weight of each of these edges is $\delta + \epsilon$, the time between the start of the loading operation of an activity and the start of the unloading operation of the next activity in π.
- Edge (v_4^l, v_0^u). The weight of this edge is $\delta + \epsilon - T$, which is a lower bound on the time between the start of the unloading operation on the first activity A_0 and the start of the loading operation of the last activity A_3.
- Edges (v_1^l, v_1^u), (v_2^l, v_2^u), and (v_3^l, v_3^u) corresponding to the loading and unloading of the part on machines $M_i, i = 1, 2, 3$, respectively. Note that activity A_0 (resp., A_2) precedes activity A_1 (resp., A_3) in π. The weight of edge (v_1^l, v_1^u) is $p_1 + \epsilon$ and the weight of edge (v_3^l, v_3^u) is $p_3 + \epsilon$; these correspond to lower bounds on the time between the start of loading and the start of unloading on machines M_1 and M_3. Since activity A_1 does not precede A_2, the weight of edge (v_2^l, v_2^u) is $\delta + \epsilon - T$, a lower bound on time between start of unloading (in the next execution of π) and loading on machine M_2.

The minimum cycle time of π equals the minimum value of T for which the graph G does not contain any cycle of positive length.

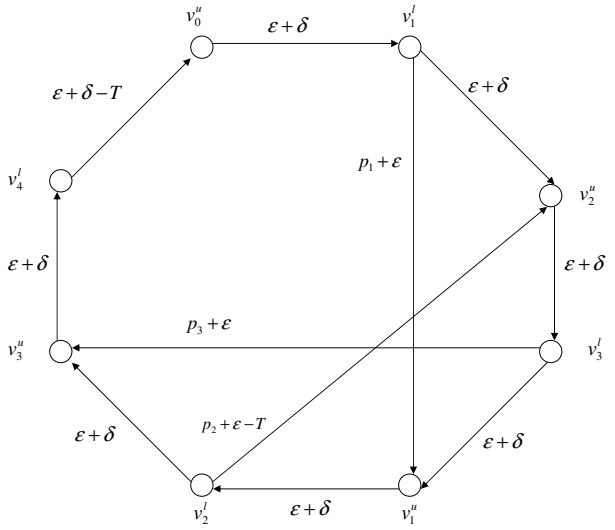


Figure 3.1. The Graph G for the Cycle of Example 3.1.

This approach can be easily extended to multi-unit cycles, as well as multiple part-types. We refer the reader to Crama et al. [39] for an extensive description.

The computation of the cycle time via the graphical approach, as well as the precise schedule of the exact time each machine is loaded and unloaded, has been studied by several authors. For free-pickup cells, Cohen et al. [36], Carlier and Chrétienne [28], Matsuo et al. [120] and van de Klundert [153] provide an $O(m^3)$ algorithm based on an algorithm of Karp [91] for finding the minimum mean-length cycle in a digraph. Other algorithms for free-pickup cells include (i) an $O(m^3)$ algorithm due to Ioachaim and Soumis [85], based on the Bellman-Ford algorithm for the longest path; (ii) an $O(m^2 \log m \log B)$ algorithm, where B is an upper bound on the optimal cycle time, based on the results of Roundy [137]; (iii) $O(m^2 \log m)$ algorithms due to Kats and Levner [94] and Lee and Posner [106], based on the Karp-Orlin algorithm [92] for finding a parametric shortest path, (iv) an $O(m^2)$ algorithm based on the results

of Hartmann and Orlin [81]. For interval cells, the results include (i) an $O(m^2 \log m \log B)$ algorithm due to Lei [107], where B is a bound on the range of possible cycle times; (ii) two algorithms based on the Bellman-Ford algorithm: an $O(m^6)$ algorithm due to Chen et al. [32] and an $O(m^3)$ algorithm due to Kats and Levner [94]. The cycle time computation for a given cycle is relatively easy in no-wait cells, and is discussed in Chapter 9.

Computing the cycle time of highly structured cycles is typically much more efficient. For example, Crama and van de Klundert [40] develop an $O(m)$ algorithm to find the cycle time in an additive travel-time cell of any member of a dominant subset of cycles called *pyramidal cycles*. Pyramidal cycles are discussed in greater detail in Section 3.3.3.

3.2.3 Lower Bounds on Cycle Times

From Equation (3.1), we can deduce a lower bound for the cycle time for a 1-unit cycle in a constant travel-time cell (problem $RF_m|$(*free,C, cyclic-1*)$|\mu$). Obviously, for any cycle, $T_c(\pi) \geq 2(m+1)\epsilon + (m+2)\delta$. If all machines with partial waiting have $w_i = 0$, then the minimum value for $T_c(\pi)$ is achieved by minimizing $\sum_{i \in V_1} p_i + \sum_{i \in V_2} \delta$, which is done by placing those machines for which $p_i \leq \delta$ in V_1. Thus, in a constant travel-time robotic cell, for any 1-unit cycle π, $T_c(\pi) \geq (m+2)\delta + \sum_{i=1}^{m} \min\{p_i, \delta\} + 2(m+1)\epsilon$ [47]. In a regular additive travel-time robotic cell (problem $RF_m|$(*free,A,cyclic-1*)$|\mu$), for any 1-unit cycle π, $T_a(\pi) \geq 2(m+1)(\delta+\epsilon) + \sum_{i=1}^{m} \min\{p_i, \delta\}$ [40].

Suppose that $p_j = \max_{1 \leq i \leq m} p_i$ is large relative to δ and ϵ. Since the cycle time can be measured as the time between successive loadings of M_j, we can derive another lower bound for the cycle time of a 1-unit cycle. This includes, at minimum, the times for the following: processing on M_j, unload M_j, move to M_{j+1}, load M_{j+1}, move to M_{j-1}, unload M_{j-1}, move to M_j, and load M_j. For constant travel time, this value is $p_j + 3\delta + 4\epsilon$ and for regular additive travel time it is $p_j + 4(\delta + \epsilon)$. We combine these bounds, originally derived by Dawande et al. [47] and Crama and van de Klundert [40], respectively, in the following theorem.

THEOREM 3.2 *For 1-unit cycles, the following are lower bounds for constant travel-time robotic cells (problem $RF_m|$(free,C,cyclic-1)$|\mu$) and regular additive travel-time robotic cells (problem $RF_m|$(free,A,cyclic-1)$|\mu$),*

respectively:

$$T_c(\pi) \geq \max\{(m+2)\delta + \sum_{i=1}^{m} \min\{p_i, \delta\} + 2(m+1)\epsilon, \max_{1\leq i\leq m} p_i + 3\delta + 4\epsilon\},$$

$$T_a(\pi) \geq \max\{2(m+1)(\delta+\epsilon) + \sum_{i=1}^{m} \min\{p_i, \delta\}, \max_{1\leq i\leq m} p_i + 4(\delta+\epsilon)\}.$$

Generalizations of these lower bounds for k-unit cycles, $k \geq 1$, are stated and proved in Theorems 3.15 and 3.19.

3.3 Optimal 1-Unit Cycles

We first examine two elementary cycles on simple robotic cells with free pickup and then examine specific conditions under which they are optimal. We then discuss two classes of cycles in which an optimal cycle can be found under more general conditions for cells with free pickup. We conclude by summarizing an approach to find an optimal cycle in no-wait cells.

3.3.1 Special Cases

In the forward cycle $\pi_U = (A_0, A_1, A_2, \ldots, A_{m-1}, A_m)$, the robot unloads a part from I, carries it to M_1, loads M_1, waits for M_1 to process the part, unloads M_1, and then carries the part to M_2. The robot continues in this fashion, waiting at each machine for its entire processing of the part. Only one machine is processing a part at any given time. A starting and ending state for this cycle is the state in which all machines are unoccupied and the robot is at the input buffer I. The processing times for π_U in constant and regular additive travel-time robotic cells, respectively, are

$$T_c(\pi_U) = 2(m+1)\epsilon + \sum_{i=1}^{m} p_i + (m+2)\delta,$$

$$T_a(\pi_U) = 2(m+1)\epsilon + \sum_{i=1}^{m} p_i + 2(m+1)\delta.$$

For constant and additive travel-time simple robotic cells, Theorems 3.3–3.6 provide an optimal 1-unit cycle under specific conditions. In terms of the classification provided in Chapter 2, these results are for problems $RF_m|(free,C,cyclic\text{-}1)|\mu$ and $RF_m|(free,A,cyclic\text{-}1)|\mu$.

THEOREM 3.3 *For both constant and regular additive travel-time robotic cells, if* $p_i \leq \delta, \forall i$, *then* π_U *achieves the optimal 1-unit cycle time.*

Proof. The result follows immediately from Theorem 3.2. ∎

The reverse cycle for a simple robotic cell is $\pi_D = (A_0, A_m, A_{m-1}, \ldots, A_2, A_1)$. To perform π_D, the robot unloads a part from the input buffer (M_0), carries it to M_1, and loads M_1. It then travels to M_m, unloads M_m, and carries that part to the output buffer (M_{m+1}). It repeats the following sequence for $i = m-1, m-2, \ldots, 1$: travel to M_i, unload M_i, carry the part to M_{i+1}, and load M_{i+1}. After loading M_2 (which completes activity A_1), the robot completes the cycle by traveling to the input buffer (M_0). At each machine, before unloading a part from it, the robot may have to wait for that machine to complete processing.

The cycle times for π_D in constant [47] and regular additive [40] travel-time robotic cells, respectively, are

$$\begin{aligned} T_c(\pi_D) &= \max\{2(m+1)(\delta+\epsilon), \max_{1\leq i\leq m} p_i + 3\delta + 4\epsilon\}, \\ T_a(\pi_D) &= \max\{4m\delta + 2(m+1)\epsilon, \max_{1\leq i\leq m} p_i + 4(\delta+\epsilon)\}. \end{aligned}$$

Note that in each expression, the first argument represents the cycle time if the robot never waits for a machine to complete its processing.

For each of the following two theorems, if its premises are met, then π_D achieves the lower bound stated in Theorem 3.2. Theorem 3.4 combines results from Dawande et al. [47] and Crama and van de Klundert [40].

THEOREM 3.4 *For the optimal 1-unit cycle problem in constant travel-time robotic cells (problem* $RF_m|(free,C,cyclic\text{-}1)|\mu$*), if* $\max_{1\leq i\leq m} p_i + 3\delta + 4\epsilon \geq 2(m+1)(\delta+\epsilon)$, *then* π_D *is an optimal 1-unit cycle. In a regular additive travel-time robotic cell (problem* $RF_m|(free,A,cyclic\text{-}1)|\mu$*), if* $\max_{1\leq i\leq m} p_i + 4(\delta+\epsilon) \geq 4m\delta + 2(m+1)\epsilon$, *then* π_D *is an optimal 1-unit cycle.*

Theorem 3.4 can be generalized to the Euclidean travel-time case (problem $RF_m|(free,E,cyclic\text{-}1)|\mu$). If

$$\max_{1\le i\le m}\{p_i+\delta_{i,i+1}+\delta_{i+1,i-1}+\delta_{i-1,i}+4\epsilon\}\ge 2(m+1)\epsilon+\sum_{i=0}^{m}\delta_{i,i+1}+\sum_{i=2}^{m+1}\delta_{i,i-2}+\delta_{1,m}, \tag{3.2}$$

then π_D is optimal. If condition (3.2) holds, then the cycle time $T_e(\pi_D)=\max_{1\le i\le m}\{p_i+\delta_{i,i+1}+\delta_{i+1,i-1}+\delta_{i-1,i}+4\epsilon\}$, which, by following the logic of Section 3.2.3 and using the triangle inequality, is a lower bound on the cycle time.

The following theorem is from Dawande et al. [47].

THEOREM 3.5 *For constant travel-time robotic cells (problem $RF_m|(free, C,cyclic\text{-}1)|\mu$), if $p_i\ge\delta$, $\forall i$, then π_D achieves the optimal 1-unit cycle time.*

This theorem does not hold for additive travel-time robotic cells. Consider the following example: $\pi_0=(A_0,A_1,A_m,A_{m-1},\ldots,A_2)$ with $p_i\ge\delta,\forall i$. The cycle time is

$$T_a(\pi_0) = \max\left\{(4m-2)\delta+2(m+1)\epsilon+p_1,\ p_2+p_1+6(\delta+\epsilon),\ \max_{3\le i\le m}\{p_i+4(\delta+\epsilon)\}\right\}. \tag{3.3}$$

If

$$\begin{aligned} p_2+p_1+6(\delta+\epsilon) &\le (4m-2)\delta+2(m+1)\epsilon+p_1, \quad \text{and}\\ \max_{3\le i\le m}\{p_i+4(\delta+\epsilon)\} &\le (4m-2)\delta+2(m+1)\epsilon+p_1, \end{aligned}$$

then $T_a(\pi_0)=(4m-2)\delta+2(m+1)\epsilon+p_1$. If $p_1<2\delta$, then $T_a(\pi_D)=4m\delta+2(m+1)\epsilon$ and $T_a(\pi_0)<T_a(\pi_D)$. However, we do have the following results, from Dawande et al. [46], for regular additive travel-time cells. Our proof of the following theorem requires specific properties of pyramidal cycles; a proof is provided in Section 3.3.3.

THEOREM 3.6 *For problem $RF_m|(free,A,cyclic\text{-}1)|\mu$, if*

$$p_i+p_{i+1}\ge(4m-6)\delta+2(m-2)\epsilon,\ i=1,\ldots,m-1,$$

then π_D is optimal.

COROLLARY 3.1 *For problem $RF_m|(free,A,cyclic\text{-}1)|\mu$, if*

$$p_i \geq (2m-3)\delta + (m-2)\epsilon,\ i = 1, \ldots, m,$$

then π_D is optimal.

In the next subsection, we provide a description of a polynomial-time algorithm to obtain an optimal 1-unit cycle in constant travel-time cells (Dawande et al. [47]). Section 3.3.3 lists the main results of a polynomial-time algorithm for additive cells due to Crama and van de Klundert [40].

3.3.2 General Cases: Constant Travel-Time Cells

We refer to the problem of finding an optimal 1-unit cycle in a constant travel-time cell as Problem Q. The results developed in Section 3.3.1 help us to identify an optimal 1-unit cycle in the following three cases:
Case 1. $p_i \leq \delta,\ \forall i$. The simple 1-unit cycle $\pi_U = (A_0, A_1, ..., A_m)$ is optimal from Corollary 3.3.
Case 2. $p_i \geq \delta,\ \forall i$. The reverse 1-unit cycle $\pi_D = (A_0, A_m, A_{m-1}, \ldots, A_1)$ is optimal from Theorem 3.5.
Case 3. $2(m+1)(\delta+\epsilon) < p_i + 3\delta + 4\epsilon$ for some $i \in M$. The reverse 1-unit cycle $\pi_D = (A_0, A_m, A_{m-1}, ..., A_1)$ is optimal from Theorem 3.4.

We label the case in which Cases 1–3 do not apply as Case 4. In this section, our aim is to characterize a class of 1-unit cycles which permits the efficient identification of an optimal 1-unit cycle in Case 4. We then develop a polynomial-time procedure FindCycle.

Basic Cycles

To establish a procedure for finding an optimal 1-unit cycle under Case 4, we first define *basic cycles*. The set of basic cycles is a subset of 1-unit cycles. Recall that V_1 is the set of machines with full waiting, and $V_2 = M \backslash V_1$ is the set of machines with partial waiting. A basic cycle corresponding to V_1 is constructed as described below. A string $S_j = \{A_j, A_{j+1}, \ldots, A_{j+\ell}\}$ is a sequence of activities such that consecutive machines M_{j+k}, $k = 1, \ldots, \ell$, have full waiting and M_j has partial waiting. The size of string S_j is $\ell+1$, where $0 \leq \ell \leq m$. Given a problem instance and V_1, strings are formed as described in the procedure below. These strings are later concatenated in a certain way to form a basic

cycle. For a given 1-unit cycle, we let n_1 denote the number of machines with full waiting and $n_2 = m - n_1$ as the total number machines with partial waiting. Thus, $n_1 = |V_1|$ and $n_2 = |V_2| = m - n_1$.

Procedure Strings

Step 0: Input: A problem instance for an m-machine robotic cell given by m, δ, ϵ, $p_1, \ldots, p_m$ and V_1. Let $j = 0$, $k = 0$ and $S_0 = \{A_0\}$.

Step 1: If $j + 1 \in V_1$, then $S_k = S_k \cup A_{j+1}$. Otherwise go to Step 3.

Step 2: $j = j + 1$. If $j < m$, then go to Step 1. Otherwise go to Step 4.

Step 3: $k = j + 1$, $j = j + 1$, and $S_k = \{A_k\}$. If $j < m$, then go to Step 1.

Step 4: Terminate.

Let there be $n_2 + 1$ strings $(S_0, S_{j_1}, \ldots, S_{j_{n_2}})$ obtained from the above procedure. The *basic cycle* corresponding to V_1 is a concatenation of $n_2 + 1$ strings in the order $S_0, S_{j_{n_2}}, S_{j_{n_2-1}} \ldots, S_{j_1}$.

REMARK 3.1 Given any 1-unit cycle, the corresponding set V_1 enables us to define the strings. For an m-machine cell, there can be multiple 1-unit cycles corresponding to a given set V_1. For example, consider $m = 4$. Then, both the cycles $\{A_0, A_3, A_1, A_2, A_4\}$ and $\{A_0, A_4, A_3, A_1, A_2\}$ have $V_1 = \{2\}$. For all the 1-unit cycles corresponding to a given set V_1, the set of strings is identical. It is the concatenation of the strings in the particular order defined above that defines a unique basic cycle corresponding to V_1. It should be noted that we have $m+1$ strings each of size one if all machines have partial waiting, whereas we have only one string of size $m + 1$ if all machines have full waiting.

EXAMPLE 3.2 $m = 8$, $V_1 = \{1, 2, 4, 8\}$ and $V_2 = \{3, 5, 6, 7\}$. We have five strings: $S_0 = \{A_0, A_1, A_2\}$, $S_3 = \{A_3, A_4\}$, $S_5 = \{A_5\}$, $S_6 = \{A_6\}$ and $S_7 = \{A_7, A_8\}$. The basic cycle corresponding to V_1 is S_0, S_7, S_6, S_5, S_3. The basic 1-unit cycle is $\pi_B = \{A_0, A_1, A_2, A_7, A_8, A_6, A_5, A_3, A_4\}$.

Next we define two sets X_i and Y_i for each $i \in V_2$. Let $V_2 = \{i_1, i_2, ..., i_{n_2}\}$ with $i_{j+1} > i_j$, $j = 1, 2, \ldots, n_2 - 1$. We also let $i_0 = 0$ and

$i_{n_2+1} = m + 1$. It is illustrative to describe these sets for Example 3.2 before we provide a formal definition. In this example, $V_2 = \{3, 5, 6, 7\}$. Note that $n_2 = 4$, $i_0 = 0$, $i_1 = 3$, $i_2 = 5$, $i_3 = 6$, $i_4 = 7$, and $i_5 = 9$. X_3 will be the set of machines with full waiting whose indices are strictly less than $i_1 = 3$ and strictly greater than $i_0 = 0$. Thus, $X_3 = \{1, 2\}$. Similarly, X_5 will be the set of machines with full waiting whose indices are strictly less than $i_2 = 5$ and strictly greater than $i_1 = 3$, and so on. Thus, $X_3 = \{1, 2\}, X_5 = \{4\}, X_6 = X_7 = \emptyset$. Now, we turn to defining the sets $Y_i, i = 3, 5, 6, 7$, for the example. Y_7 will be the set of those machines with full waiting whose indices are strictly greater than $i_4 = 7$ and strictly less than $i_5 = 9$. Similarly, Y_6 will be the set of machines with full waiting whose indices are strictly greater than $i_3 = 6$ and strictly less than $i_4 = 7$, and so on. Thus, $Y_7 = \{8\}$, $Y_6 = Y_5 = \emptyset$ and $Y_3 = \{4\}$. These sets are illustrated in Figure 3.2.

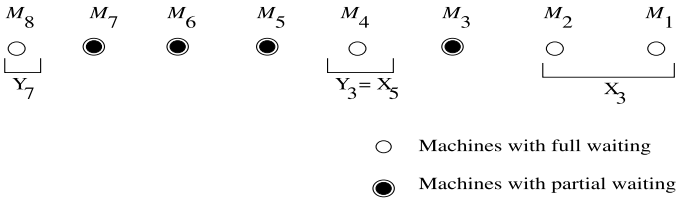


Figure 3.2. Sets X_i and Y_i for $i \in V_2$ in Example 3.2.

We are now ready to define the sets X_i and Y_i, $i \in V_2$, in the general case.

DEFINITION 3.6 Suppose $V_2 = \{i_1, i_2, ..., i_{n_2}\}$ with $i_{j+1} > i_j, j = 1, 2, ..., n_2 - 1$. We also let $i_0 = 0$ and $i_{n_2+1} = m+1$. Then for $i_j, j = 1, 2, ..., n_2$,

$$\begin{aligned} X_{i_j} &= \{i_{j-1} < k < i_j : k \in V_1\}, j = 1, 2, \ldots, n_2, \\ Y_{i_j} &= \{i_{j+1} > k > i_j : k \in V_1\}, j = 1, 2, \ldots, n_2. \end{aligned}$$

In terms of the strings defined above, the sets X_i and Y_i can be expressed as follows: X_i is the set of machines with full waiting associated with the string that immediately follows S_i in the basic cycle corresponding to V_1, and Y_i is the set of machines with full waiting associated with the string S_i. We also denote $|X_i| = r_i$ and $|Y_i| = q_i$.

THEOREM 3.7 *For a basic 1-unit cycle π_B, the cycle time $T(\pi_B)$ can be expressed as follows:*

$$T(\pi_B) = \max\{\alpha, \beta_i | i \in V_2\}, \tag{3.4}$$

where

$$\alpha = 2(m+1)\delta + 2(m+1)\epsilon - n_1\delta + \sum_{i \in V_1} p_i, \tag{3.5}$$

$$\beta_i = p_i + 3\delta + 4\epsilon + r_i(\delta + 2\epsilon) + \sum_{j \in X_i} p_j + q_i(\delta + 2\epsilon) + \sum_{j \in Y_i} p_j. \tag{3.6}$$

Proof. The cycle time $T(\pi_B)$ is the sum of (i) the total robot move time t_m, (ii) the total load/unload time t_l, and (iii) the total robot wait time at machines in a cycle. The total robot wait time is the sum of two components: the total partial waiting time W_p and the total full waiting time W_f. Thus, $T(\pi_B) = t_m + t_l + W_f + W_p$. Note that $t_m + t_l = 2(m+1)\delta + 2(m+1)\epsilon - n_1\delta$ and $W_f = \sum_{i \in V_1} p_i$. Thus, we have $\alpha = t_m + t_l + W_f$,

$$T(\pi_B) = \alpha + W_p, \tag{3.7}$$

and $W_p = \sum_{i \in V_2} w_i$, where $w_i = \max\{0, p_i - t_i\}$ and t_i denotes the elapsed time between the moment the robot completes loading a part on M_i and the moment the robot returns to the machine M_i for unloading the part. Note that $t_i = T(\pi_B) - w_i - \Delta_i$, where Δ_i is the elapsed time between the moment the robot begins to unload a part from machine M_i and the moment the robot completes loading a part on M_i during a cycle. Δ_i is the sum of the following times: the time $(\delta + 2\epsilon + r_i(\delta + 2\epsilon) + \sum_{j \in X_i} p_j)$ elapsed between unloading machine M_i and loading the machine in V_2 immediately following M_i, the time (δ) to travel to the machine in V_2 immediately preceding M_i, the waiting time at this machine to finish processing, and the time $(\delta + 2\epsilon + q_i(\delta + 2\epsilon) + \sum_{j \in Y_i} p_j)$ elapsed between unloading this machine and loading M_i. More precisely, the expression for Δ_i can be written as follows:

$$\Delta_i = 3\delta + 4\epsilon + r_i(\delta + 2\epsilon) + \sum_{j \in X_i} p_j + q_i(\delta + 2\epsilon) + \sum_{j \in Y_i} p_j + w_{k_i}.$$

Thus,

$$t_i = W_p + \alpha - w_i - [3\delta + 4\epsilon + r_i(\delta + 2\epsilon) + \sum_{j \in X_i} p_j + q_i(\delta + 2\epsilon) + \sum_{j \in Y_i} p_j] - w_{k_i},$$

where w_{k_i} is the partial waiting encountered by the robot at machine M_{k_i}, where k_i is defined as follows. If $\{j \in V_2 : j < i\} \neq \emptyset$, then $k_i = \max\{j \in V_2 : j < i\}$. Otherwise, $k_i = 0$ and $w_0 = 0$. Note that $k_i = 0$ iff i is the first index in V_2. Thus, we have

$$\begin{aligned} w_i &= \max\Bigg\{0, p_i - W_p - \alpha + [3\delta + 4\epsilon + r_i(\delta + 2\epsilon) \sum_{j \in X_i} p_j \\ &\qquad + q_i(\delta + 2\epsilon) + \sum_{j \in Y_i} p_j] + w_i + w_{k_i}\Bigg\} \\ &= \max\{0, \beta_i - \alpha - W_p + w_i + w_{k_i}\}. \end{aligned} \tag{3.8}$$

It follows from (3.8) that

$$W_p \geq \beta_i - \alpha + w_{k_i} \geq \beta_i - \alpha, \qquad \forall\ i \in V_2. \tag{3.9}$$

Case 1. $W_p = 0$: Then, $w_i = 0$, $\forall\ i \in V_2$. It follows from (3.8) that $\forall\ i \in V_2, \beta_i - \alpha \leq 0$ and hence $W_p = 0 = \max\{0, \beta_i - \alpha, i \in V_2\}$.
Case 2. $W_p > 0$: Let $i^* = \min\{i \in V_2 : w_i > 0\}$. Note that $w_{k_{i^*}} = 0$. Also from (3.8), it follows that $W_p = \beta_{i^*} - \alpha$. Using (3.9), we have $W_p = \max\{\beta_i - \alpha, i \in V_2\} = \max\{0, \beta_i - \alpha, i \in V_2\}$.

Thus, we have

$$W_p = \max\{0, \beta_i - \alpha, i \in V_2\}. \tag{3.10}$$

The result follows from (3.7) and (3.10). ∎

REMARK 3.2 There does not exist a 1-unit cycle with $|V_2| = 1$. Otherwise, for exactly one machine, say M_r, the robot loads M_r and leaves it to travel to some other machine. Since the robot must remain with the part at all other machines, it could never return to pick up the part at M_r. Thus, if $V_2 \neq \emptyset$, then $|V_2| \geq 2$.

In general, there may be many cycles corresponding to a given V_1. However, there is a unique basic cycle among them. We show below

that this basic cycle dominates the other cycles. To show this result, we need to define another Problem R of finding an optimal 1-unit cycle under the following cell data:

- Processing time p_i on machine M_i, $i = 1, ..., m$.
- A constant travel time δ from machine M_i to M_j, $i \neq j$, when the robot travels empty.
- A constant travel time δ_i from machine M_{i-1} to M_i, when the robot travels with a part and $\delta_i \geq \delta$, $1 \leq i \leq m+1$.
- A constant loading and unloading time ϵ.

LEMMA 3.2 *For Problem* R, *the reverse cycle dominates all other cycles for which* $V_1 = \emptyset$.

Proof. For Problem R, the expression for the reverse 1-unit cycle $\pi_D = (A_0, A_m, A_{m-1}, \ldots, A_1)$ is

$$T^R(\pi_D) = \max \left\{ \sum_{k=1}^{m+1} \delta_k + (m+1)\delta + 2(m+1)\epsilon, \right. \\ \left. p_i + \delta + \sum_{k=i}^{i+1} \delta_k + 4\epsilon \mid i \in V_2 \right\}.$$

Note that for *any* permutation π corresponding to $V_1 = \emptyset$, we have $T^R(\pi) \geq p_i + \delta + \sum_{k=i}^{i+1} \delta_k + 4\epsilon$. After unloading machine M_i, the robot loads M_{i+1} and later returns to load machine M_i. This requires at least $\delta + \sum_{k=i}^{i+1} \delta_k + 4\epsilon$ time. Also, the robot has to process the part on machine M_i exactly once. This requires time p_i. Thus, $T^R(\pi) \geq \max_{1 \leq i \leq m} \{p_i + \delta + \sum_{k=i}^{i+1} \delta_k + 4\epsilon\}$. Next, for π with $V_1 = \emptyset$, we have $T^R(\pi) \geq \sum_{k=1}^{m+1} \delta_k + (m+1)\delta + 2(m+1)\epsilon$. Thus $T^R(\pi) \geq T^R(\pi_D)$ and the result follows. ∎

THEOREM 3.8 *For Problem* Q, *basic cycles dominate all other 1-unit cycles.*

Proof. Consider any *nonbasic* 1-unit cycle π' corresponding to V_1. Let π be the unique basic 1-unit cycle corresponding to V_1. Let $V_2 = \{i_1, i_2, \ldots, i_{n_2}\}$. We show that $T(\pi) \leq T(\pi')$. Construct the following instance I of Problem R: We have n_2 machines $M_1', M_2', ..., M_{n_2}'$,

with the processing time at machine $M_k^{'} = p_{i_k}$, $k = 1, 2, \ldots, n_2$; $\delta_k = \delta + \sum_{j=i_{k-1}+1}^{i_k-1} p_j + (i_k - i_{k-1} - 1)(\delta + 2\epsilon)$, $k = 1, 2, \ldots, n_2 + 1$. Recall that $i_0 = 0$. Thus, all 1-unit cycles of instance I are equivalent to permutations of the strings of Problem Q given the set V_2. We note the following equivalence between Problem Q and Problem R:

1. The cycle time $T(\pi)$ of the basic cycle π for Problem Q is the same as that of the reverse cycle in instance I for Problem R. This follows from equations (3.4)–(3.6) and the formula for $T^R(\pi_D)$ in the proof of Lemma 3.2.

2. The cycle time $T(\pi^{'})$ of the nonbasic cycle $\pi^{'}$ for Problem Q is the same as that of a cycle in instance I corresponding to $\bar{V}_1 = \emptyset$ and $\bar{V}_2 = \{1, 2, ..., n_2\}$. This follows since the time required by $\pi^{'}$ to complete the processing on machines in V_1 is accounted for in instance I using the travel times δ_k.

The result now follows immediately from Lemma 3.2. ∎

Theorem 3.8 allows us to focus only on basic 1-unit cycles to look for an optimal 1-unit cycle. In the remainder of this section, we further characterize an optimal 1-unit (basic) cycle for Problem Q. The characterization leads to a subclass of the class of basic cycles containing an optimal 1-unit cycle.

To obtain these characterization results, we define the set D_δ of machines on which the processing time is at least the travel time δ, i.e.,

$$D_\delta = \{i \in M : p_i \geq \delta\}.$$

THEOREM 3.9 *If* $|D_\delta| = 1$ *and* $p_i + p_j \leq 2\delta$, $\forall i, j$, *then the simple 1-unit cycle* $\pi_U = (A_0, A_1, ..., A_m)$ *is an optimal 1-unit cycle.*

Proof. The cycle time of the simple 1-unit cycle π_U is $T(\pi_U) = (m + 2)\delta + \sum_{i=1}^{m} p_i + 2(m+1)\epsilon$. Consider an optimal 1-unit cycle $\pi' \neq \pi_U$. Let V_1 and V_2 be the sets of machines with full and partial waiting, respectively, in $\pi^{'}$. We may assume that $\pi^{'}$ is a basic cycle due to Theorem 3.8. Since $|D_\delta| = 1$, let i^* be the unique machine with $p_{i^*} \geq \delta$. Since $\pi^{'} \neq \pi_U$, we have $V_2 \neq \emptyset$. Note that $|V_2| \geq 2$. From Theorem 3.7, $T(\pi^{'}) \geq [2(m+1) - n_1]\delta + \sum_{i \in V_1} p_i + 2(m+1)\epsilon = (m+2)\delta + \sum_{i \in V_2}(\delta - p_i) + \sum_{i=1}^{m} p_i + 2(m+1)\epsilon$. Now if $i^* \in V_1$, then $p_i \leq \delta$, $\forall i \in V_2$, and

hence $\sum_{i\in V_2}(\delta - p_i) \geq 0$. Consequently, $T(\pi^{'}) \geq T(\pi_U)$. If $i^* \in V_2$ and since $|V_2| \geq 2$, there exists a machine $M_j \neq M_{i^*}$ with $j \in V_2$. By the hypothesis, $p_{i^*} + p_j \leq 2\delta$. Then $T(\pi^{'}) = (m+2)\delta + (2\delta - p_{i^*} - p_j) + \sum_{i\in V_2\backslash\{i^*,j\}}(\delta - p_i) + \sum_{i=1}^{m} p_i + 2(m+1)\epsilon \geq T(\pi_U)$. ∎

THEOREM 3.10 *If $|D_\delta| \geq 2$, then there exists an optimal basic 1-unit cycle with $V_2 \neq \emptyset$.*

Proof. Suppose not. Since the simple 1-unit cycle π_U is the only cycle with $V_2 = \emptyset$, let it be the unique optimum 1-unit cycle. Since $|D_\delta| \geq 2$, there exist $i, j \in D_\delta$. Consider the basic 1-unit cycle π corresponding to $V_2 = \{i, j\}$ and $V_1 = M\backslash V_2$. From Theorem 3.7, it follows that $T(\pi) = \max\{\alpha^*, \beta_i^*, \beta_j^*\}$ with $\alpha^* \leq T(\pi_U)$ and $\beta_k^* \leq T(\pi_U)$ for $k = i, j$. Thus $T(\pi) \leq T(\pi_U)$, which is a contradiction. ∎

THEOREM 3.11 *Consider the following three cases:*

1. $|D_\delta| = 0$: *The simple 1-unit cycle $\pi_U = (A_0, A_1, ..., A_m)$ is an optimal 1-unit cycle.*

2. $|D_\delta| \geq 2$: *There exists an optimal basic cycle in which $D_\delta \subseteq V_2$.*

3. $|D_\delta| = 1$: *Let $p_q \geq \delta$ and $p_k = \max\{p_j : j \in M\backslash\{q\}\}$. If $p_q + p_k \leq 2\delta$, then the simple 1-unit cycle $\pi_U = (A_0, A_1, ..., A_m)$ is an optimal 1-unit cycle. If $p_q + p_k \geq 2\delta$, then there exists an optimal basic 1-unit cycle in which $q \in V_2$.*

Proof.

Case 1. Follows from Corollary 3.3.

Case 2. Suppose for machine M_j, we have $p_j \geq \delta$ and $j \in V_1$ in an optimal 1-unit cycle π. Without loss of generality, we can assume $V_2 \neq \emptyset$ due to Theorem 3.10. From Theorem 3.7, $T(\pi) = \max\{\alpha, \beta_i, i \in V_2\}$. Let $\xi_j^1 = \{k : k < j, k \in V_2\}$ and $\xi_j^2 = \{k : k > j, k \in V_2\}$. Note that at least one of ξ_j^1 or ξ_j^2 is non-empty. If $\xi_j^1 \neq \emptyset$, let $j_1 = \max \xi_j^1$, and if $\xi_j^2 \neq \emptyset$, let $j_2 = \min \xi_j^2$. We change the waiting at M_j from full to partial. Define $V_2^{'} = V_2 \cup \{j\}$ and $V_1^{'} = V_1\backslash\{j\}$. Let $\pi^{'}$ be the basic 1-unit cycle corresponding to $(V_1^{'}, V_2^{'})$. Then $T(\pi^{'}) = \max\{\alpha^{'}, \beta_i^{'}, i \in V_2^{'}\}$. Also, $\alpha^{'} = \alpha + \delta - p_j \leq \alpha$, $\beta_{j_1}^{'} < \beta_{j_1}$ (if j_1 exists), $\beta_{j_2}^{'} < \beta_{j_2}$ (if j_2 exists), $\beta_j^{'} \leq \beta_{j_l}$ for $l = 1, 2$ and $\beta_l^{'} = \beta_l$ for $l \in V_2\backslash\{j_1, j_2\}$. Thus $T(\pi^{'}) \leq T(\pi)$.

Case 3. If $p_q + p_k \leq 2\delta$, then the optimality of π_U follows from Theorem 3.9. If $p_q + p_k > 2\delta$, observe that there exists an optimal solution $\pi \neq \pi_U$. For otherwise, if π_U is the unique optimal solution, then the basic cycle corresponding to $V_2 = \{q, k\}$ and $V_1 = M \backslash V_2$ will satisfy $T(\pi) \leq T(\pi_U)$, which is a contradiction to the uniqueness of π_U. Now consider an optimal solution $\pi^{'} \neq \pi_U$, where $\pi^{'}$ is the basic cycle corresponding to $(V_1^{'}, V_2^{'})$ with $q \in V_1^{'}$. Note that $|V_2^{'}| \geq 2$. Define $V_2^* = V_2^{'} \cup \{q\}$ and $V_1^* = M \backslash V_2^*$. Let π^* be the basic 1-unit cycle corresponding to (V_1^*, V_2^*). The proof of $T(\pi^*) \leq T(\pi^{'})$ is similar to that in Case 2. ∎

Theorem 3.11 offers a fundamental insight which is easy to state: If $D_\delta \neq \emptyset$, we can assume $D_\delta \subseteq V_2$. We use this property to construct an initial partition of the set of machines into those where the robot has full waiting and where it has partial waiting. Recall from Corollary 3.3 that if $D_\delta = \emptyset$, the simple 1-unit cycle $\pi_U = (A_0, A_1, ..., A_m)$ is optimal. The following definition, therefore, assumes that $|D_\delta| \geq 1$.

DEFINITION 3.7 An *Initial Partition* $F = (V_1, V_2)$ is a partition of the set of machines M into two subsets V_1 and V_2 corresponding to machines with full waiting and partial waiting. These are defined as follows:

1. If $|D_\delta| \geq 2$, then define $V_2 = D_\delta$ and $V_1 = M \backslash D_\delta$.
2. If $|D_\delta| = 1$ and $p_q + p_k > 2\delta$, where $p_q \geq \delta$, $p_k = \max\{p_j : j \in M \backslash \{q\}\}$, then define $V_2 = \{q\}$ and $V_1 = M \backslash V_2$. Note that if $p_q + p_k \leq 2\delta$, then π_U is optimal (Theorem 3.9).

3.3.2.1 Optimization over Basic Cycles

In this section, we develop a polynomial-time algorithm to find an optimum solution to problem instances under Case 4 (as defined at the start of Section 3.3.2). For ease of reference, we refer to this algorithm as FindCycle. First, we describe a polynomial-time solution to the decision question corresponding to the optimization problem. The use of this polynomial-time solution in a binary search procedure gives the algorithm FindCycle.

Since only basic cycles are considered, we use the cycle time expression of Theorem 3.7. Under Case 4, the cycle time $T(\pi^*)$ of an optimal 1-unit cycle π^* satisfies $2(m+1)\delta - n_1\delta + \sum_{i \in M \backslash D_\delta} p_i + 2(m+1)\epsilon \leq T(\pi^*) \leq 2(m+1)\delta + 2(m+1)\epsilon$. Note that the lower bound follows from

Theorem 3.2, while the upper bound is the cycle time of the reverse cycle π_D under Case 4. Consider the following question:

Decision Problem (DQ): Given B with $2(m+1)\delta - n_1\delta + \sum_{i \in M \setminus D_\delta} p_i + 2(m+1)\epsilon \leq B \leq 2(m+1)\delta + 2(m+1)\epsilon$, does there exist a 1-unit cycle π with $T(\pi) \leq B$?

In the remainder of this section, we will describe a polynomial-time algorithm to answer DQ. Given an initial partition $F = (V_1, V_2)$, we define the following structure:

DEFINITION 3.8 A *chain* of length n is a maximal sequence of consecutive machine indices $< i, i+1, \ldots, i+n >$ satisfying the following conditions:

1. Either $(i = 1)$ or $(i \in V_2$ and $i - 1 \in V_2)$.
2. Either $(i + n = m)$ or $(i + n \in V_2$ and $i + n + 1 \in V_2)$.
3. No two consecutive indices belong to V_2. At least one index belongs to V_1.

EXAMPLE 3.3 $m = 15$. Consider an initial partition $F = (V_1, V_2)$, where $V_1 = \{3, 7, 9, 10, 11, 13, 15\}$ and $V_2 = \{1, 2, 4, 5, 6, 8, 12, 14\}$. Then we have the following two chains (see Figure 3.3):

1. $C_1 = \ < 2, 3, 4 >$.
2. $C_2 = \ < 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 >$.

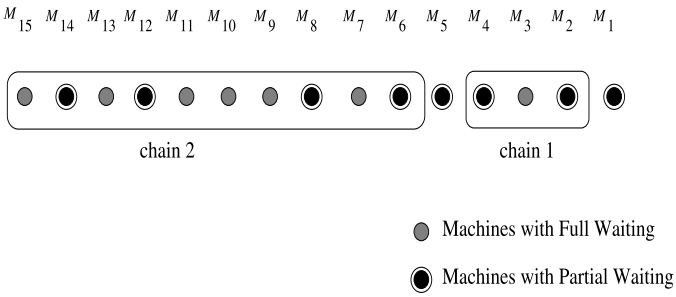


Figure 3.3. Chains Corresponding to an Initial Partition $F = (V_1, V_2)$.

Given an initial partition $F = (V_1, V_2)$, the chains are mutually exclusive and every element of V_1 is contained in exactly one chain. Let C_r, $r =$

$1, ..., c$, be the chains for F and let $\mathcal{C} = \{C_r : r = 1, ..., c\}$. We solve one shortest path problem for each $C_r \in \mathcal{C}$ to answer the decision problem DQ for a given value of B.

A Shortest Path Problem on a Chain:

First we briefly state the intuition behind the solution described below. The set of machines with partial waiting uniquely determines the corresponding basic cycle. From Theorem 3.11, we know that there exists an optimal solution in which the machines in the set D_δ have partial waiting. Thus, to completely determine the set of machines with partial waiting, we need to determine which machines, if any, from $M \backslash D_\delta$ have partial waiting. In this context, we use the shortest path problem to optimally determine those machines.

Observe that each chain can be considered independently. As mentioned above, each element of V_1 is contained in exactly one chain. Let $k \in V_1$ be contained in chain C_r. Changing the status of machine M_k from full waiting to partial waiting affects only the β_i terms in (3.4) corresponding to the two machine indices in V_2 closest to k on either side in chain C_r. By the definition of a chain, such an index belongs to the same chain C_r. More precisely, the terms corresponding to the following machine indices (if they exist) will be affected in the cycle time expression: (i) $\max\{j : j < k, j \in V_2 \cap C_r\}$ and (ii) $\min\{j : j > k, j \in V_2 \cap C_r\}$. A new β_k term, corresponding to machine M_k, will now be part of the cycle time expression. However, as will be shown shortly, this additional term will not dominate the existing terms in the cycle time expression (3.4) and hence need not be explicitly stated. We illustrate this using Example 3.3 above. Consider $k = 10 \in V_1$ contained in chain C_2. If the status of machine M_{10} is changed from full waiting to partial waiting, the only existing β_i terms in (3.4) that change are β_8 and β_{12} corresponding to machines M_8 and M_{12}, respectively.

Before we proceed with the description of the shortest path problem, we note the connection between strings (defined at the start of Section 3.3.2) and chains (defined in Section 3.3.2.1). Note that a string is a sequence of consecutive activities, while a chain is a sequence of consecutive machine indices. Consider a string $S_j = \{A_j, A_{j+1}, ..., A_{j+l}\}$ with $l \geq 1$. Then, the machines $M_{j+1}, ..., M_{j+l}$, belong to the same chain. Strings that have exactly one activity do not belong to any chain and

are responsible for the chain decomposition of the machine indices. It is also instructive to examine the changes in the strings when the status of a machine changes from full waiting to partial waiting. Consider a machine $M_i, i \in V_1$. Then, the activity subsequence $\{A_{i-1}, A_i\}$ appears in a unique string, say S_k. When the status of M_i changes from full waiting to partial waiting, the string S_k decomposes into two substrings: the substring of S_k up to and including activity A_{i-1}, and the substring of S_k containing A_i and subsequent activities.

As mentioned before, the purpose of the shortest path problem is to determine the machines, if any, from $M \backslash D_\delta$ which have partial waiting. Recall from Theorem 3.7 that the cycle time expression of a basic 1-unit cycle π can be written as $T(\pi) = \max\{\alpha, \beta_l | l \in V_2\}$. For each chain $C_r \in \mathcal{C}$, where $\mathcal{C}$ is the set of chains obtained from the initial partition $F = (V_1, V_2)$ as described above, we construct a directed graph G^r whose node set corresponds to the set of machines in $(M \backslash D_\delta) \cap C_r$ plus two distinguished nodes s and t. Given a target cycle time B by the decision problem DQ, the graphs are constructed in such a way that there exists a basic cycle with cycle time at most B if and only if there exists an s-t path in *each* graph G^r, $r = 1, 2, ..., c$, such that the sum of the lengths of these paths is at most $B - \alpha$. Moreover, the nodes on the s-t paths specify the machines from $M \backslash D_\delta$ that have partial waiting in the resulting basic cycle.

Consider a chain $C_r \in \mathcal{C}$. Let $\hat{V}_1$ and $\hat{V}_2$ be the sets of machine indices in C_r with full and partial waiting, respectively. We construct a weighted directed graph $G^r(V^r, E^r)$ as follows:

The node set consists of the node indices in $\hat{V}_1$ plus two dummy nodes s and t. That is, $V^r = \hat{V}_1 \cup \{s, t\}$. Let $E_1^r = \{(s, j) : j \in \hat{V}_1\}, E_2^r = \{(l, m) : l, m \in \hat{V}_1\}$, and $E_3^r = \{(k, t) : k \in \hat{V}_1\}$. The edge set E^r is either the single edge (s, t) or equals $\bar{E}^r = \bar{E}_1^r \cup \bar{E}_2^r \cup \bar{E}_3^r$, where $\bar{E}_k^r \subseteq E_k^r$, $k = 1, 2, 3$. Before we proceed with the precise description of $\bar{E}^r$, we explain the main ideas behind its construction.

Changing the status of a machine $M_i, i \in M \backslash D_\delta$, from full waiting to partial waiting affects the existing terms in the cycle time expression (3.4) as follows: (i) the first term α increases by $\delta - p_i$ and (ii) the terms $\beta_l, l \in V_2$, either decrease or stay unchanged. A new term β_i corresponding to machine M_i is now part of the cycle time expression.

However, this additional term will not dominate the existing terms in the cycle time expression: Using (3.6) and the properties (i) $r_i + q_i \leq m-2$ and (ii) $p_j \leq \delta, j \in X_i \cup Y_i$, it is easy to verify that $\beta_i \leq \sum_{j \in M \setminus D_\delta} p_j + (m+1)\delta + 2m\epsilon$ and is therefore strictly less than the lower bound on the cycle time established in Theorem 3.2. Thus, this additional term, β_i, need not be explicitly stated in the cycle time expression.

The edge set $\bar{E}^r$ consists of three types of edges: (a) **source edges** $(s,i), i \in \hat{V}_1$; (b) **transition edges** $(i,j), i,j \in \hat{V}_1, i < j$; and (c) **sink edges** $(i,t), i \in \hat{V}_1$. Each edge represents changing the status of a machine $M_i, i \in \hat{V}_1$, from full waiting to partial waiting. Since each element of $\hat{V}_1$ is contained in exactly one chain, to answer the decision problem DQ we need to examine exactly one chain to determine whether an element in $\hat{V}_1$ should be moved to $\hat{V}_2$.

The length of an s-t path (if one exists) in G^r measures the total increase in the first term α when the status of all the machines corresponding to the nodes on the path changes from full waiting to partial waiting. Given a target cycle time B, if there exists an s-t path in each graph $G^r, r = 1, ..., c$, such that the sum of the lengths of these paths is at most $B - \alpha$, then the status of the machines corresponding to the nodes on the paths can be changed from full waiting to partial waiting to obtain a basic 1-unit cycle with cycle time at most B.

We now describe the construction of $\bar{E}^r_k,\ k = 1, 2, 3$. To describe the construction steps, we define, for sets $X_a, Y_b \subseteq \hat{V}_1$ and a machine index $i \in \hat{V}_2$, the quantity $\beta'_i(X_a, Y_b) = p_i + 3\delta + 4\epsilon + |X_a|(\delta + 2\epsilon) + \sum_{k \in X_a} p_k + |Y_b|(\delta + 2\epsilon) + \sum_{k \in Y_b} p_k$. An explanation is provided immediately following each construction step.

1. **Construction of dummy edge:** Add edge (s,t) to E^r with edge length $d_{st} = 0$ if $|D_\delta \cap C_r| \geq 2$ and $\beta_k \leq B$ for all $k \in \hat{V}_2$ and stop construction. Otherwise, do the following construction steps.

   Explanation: Choosing edge (s,t) in G^r corresponds to making no changes in the waiting status for machines $M_k, k \in C_r$. The only terms in the cycle time expression (3.4) corresponding to machines in C_r are $\beta_k, k \in \hat{V}_2$. Since $\beta_k \leq B,\ \forall\ k \in \hat{V}_2$, there is no need to change the status of any machine.

As mentioned previously, chains can be considered independently. Therefore, changes in the waiting status of machines belonging to other chains will not affect the terms in (3.4) corresponding to machines in C_r.

Since there is no change in the status for any machine in C_r, there is no increase in the first term α of (3.4). Therefore, the length d_{st} of edge (s,t) is 0.

2. **Construction of $\bar{E}_1^r$ (source edges):** Consider $j \in V^r \backslash \{s,t\}$. Let $e = \min\{k : k \in \hat{V}_2\}$. If $j < e$, then we add edge (s,j) with length $d_{sj} = \delta - p_j$ to $\bar{E}_1^r$. Otherwise, let $H_j = \{k : k \in \hat{V}_2, k < j\}$, $q = \max\{k : k \in H_j\}$, and $\hat{Y}_q = \{k : q < k < j\}$. We add edge (s,j) with length $d_{sj} = \delta - p_j$ to $\bar{E}_1^r$ if, and only if, $\beta_q'(X_q, \hat{Y}_q) \leq B$ and $\beta_k \leq B \ \forall \ k \in H_j \backslash \{q\}$.

   Explanation: Choosing edge (s,j) corresponds to changing the status of machine M_j from full waiting to partial waiting. The construction ensures that each term $\beta_i, i \leq j, i \in C_r$, in the cycle time expression (3.4) has value at most B. The set H_j corresponds to those machines with partial waiting whose indices are strictly less than j. If edge (s,j) is chosen, the value of the term in (3.4) corresponding to machine M_q, where q is the highest index in H_j, will change. Note that if edge (s,j) is chosen, the only terms $\beta_i, i \leq j, i \in C_r$, in (3.4) are $\beta_i, i \in H_j \cup \{j\}$. We investigate the values of these β_i terms in (3.4) as a consequence of choosing edge (s,j):

   (a) As explained above, a new term β_j, corresponding to machine M_j, will be part of (3.4). This term has value less than the lower bound on the cycle time established in Theorem 3.2 and hence can be ignored.

   (b) The value of the term corresponding to machine M_q will change to $\beta_q'(X_q, \hat{Y}_q)$.

   (c) The values of terms corresponding to machines $M_k, k \in H_j \backslash \{q\}$, will remain β_k.

   Thus, given a target value of B, the edge (s,j) is constructed iff each term $\beta_i, i \in H_j \cup \{j\}$, has value at most B. The length d_{sj} of edge (s,j) is the amount of increase in the first term α of (3.4) if the status

of M_j changes from full waiting to partial waiting and equals $\delta - p_j$ as explained earlier.

3. **Construction of $\bar{E}_2^r$ (transition edges):** Let $i, j \in V^r \backslash \{s, t\}$ with $j > i$. Let $H_j = \{l : l \in \hat{V}_2, i < l < j\}$. We consider three cases:

   (a) $H_j = \emptyset$: Add edge (i, j) to $\bar{E}_2^r$ with length $d_{ij} = \delta - p_j$.

   (b) $|H_j| = 1$: Let $H_j = \{q\}$. Also let $\hat{X}_q = \{l \in \hat{V}_1 : i < l < q\}$ and $\hat{Y}_q = \{l \in \hat{V}_1 : q < l < j\}$. Include edge (i, j) with length $d_{ij} = \delta - p_j$ in $\bar{E}_2^r$ iff $\beta_q^{'}(\hat{X}_q, \hat{Y}_q) \leq B$.

   (c) $|H_j| \geq 2$: Let $q = \max\{k : k \in H_j\}$ and $v = \min\{k : k \in H_j\}$. Let $\hat{Y}_q = \{k : q < k < j\}$ and $\hat{X}_v = \{k : i < k < v\}$. Add edge (i, j) with length $d_{ij} = \delta - p_j$ to $\bar{E}_2^r$ iff $\beta_q^{'}(X_q, \hat{Y}_q) \leq B$, $\beta_v^{'}(\hat{X}_v, Y_v) \leq B$, and $\beta_k \leq B \; \forall \; k \in H_j \backslash \{q, v\}$.

   Explanation: Choosing edge (i, j) corresponds to changing the status of machine M_j from full waiting to partial waiting. The construction step ensures that an edge $(i, j), i, j \in V^r \backslash \{s, t\}$ with $j > i$, is added to the graph G^r iff each term $\beta_k, i < k \leq j$, in (3.4) has value at most B. If M_j changes status from full waiting to partial waiting, then the terms corresponding to machines M_q and M_v change to $\beta_q^{'}(X_q, \hat{Y}_q)$ and $\beta_v^{'}(\hat{X}_v, Y_v)$, respectively. The terms corresponding to machines $M_k, k \in H_j \backslash \{q, v\}$, will remain β_k. The explanation of the values of β_j and d_{ij} is similar to that provided in the previous step.

4. **Construction of $\bar{E}_3^r$ (sink edges):** Let $j \in V^r \backslash \{s, t\}$ be such that either $(s, j) \in \bar{E}_1^r$ or $(i, j) \in \bar{E}_2^r$ for some $i \in V^r \backslash \{s, t\}$. Let $f = \max\{k : k \in \hat{V}_2\}$. If $j > f$, then we add edge (j, t) with length $d_{jt} = 0$ to $\bar{E}_3^r$. Otherwise, let $H_j = \{l : l \in \hat{V}_2, l > j\}$, $q = \min\{k : k \in H_j\}$, and $\hat{X}_q = \{k : j < k < q\}$. We add edge (j, t) with length $d_{jt} = 0$ to $\bar{E}_3^r$ iff $\beta_q^{'}(\hat{X}_q, Y_q) \leq B$ and $\beta_k \leq B \; \forall \; k \in H_j \backslash \{q\}$.

   Explanation: The existence of edge (j, t) ensures that each term $\beta_i, i > j, i \in C_r$, in the cycle time expression (3.4) has value at most B. Note that edge (j, t) does not correspond to changing the status of any machine. If $j > f$, then (3.4) does not contain any term $\beta_i, i > j$. Otherwise, the terms $\beta_j, j \in H_j$ appear in (3.4) and edge (j, t) is added iff all these terms have value at most B.

Since edge (j,t) does not correspond to changing the waiting status of any machine, its length $d_{jt} = 0$.

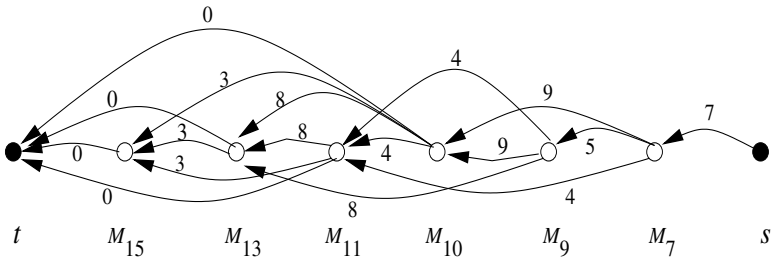


Figure 3.4. Graph G^2 Corresponding to Chain C_2 in Example 3.3.

EXAMPLE 3.3, CONTINUED: We illustrate the construction above on the graph G^2 corresponding to chain C_2. We assume the following data:

- $\epsilon = 1$, $\delta = 10$.
- The vector of processing times $(276, 275, 4, 280, 286, 299, 3, 255, 5, 1, 6, 266, 2, 245, 7)$, where the ith entry corresponds to the processing time p_i on machine M_i.

The values of $\alpha, \beta_i,\ i \in V_2$, as defined in Theorem 3.7, are as follows: $\alpha = 310, \beta_1 = 310, \beta_2 = 325, \beta_4 = 330, \beta_5 = 320, \beta_6 = 348, \beta_8 = 352, \beta_{12} = 362$, and $\beta_{14} = 312$. Thus, the cycle time is $T(\hat{\pi}) = \max\{\alpha, \beta_i | i \in V_2\} = 362$.

Consider the following instance of DQ: *Does there exist a 1-unit cycle* π *with* $T(\pi) \leq 334 = B$?

Since $\beta_i \leq 334, \forall i \in C_1 \cap V_2$, the graph G^1 will contain only the dummy edge (s,t). Therefore, we need to consider the shortest s-t path problem only on the chain C_2. Figure 3.4 shows the corresponding graph G^2.

Consider any s-t path $s - i_1 - i_2 - ... - i_l - t$ in G^r. If the status of the machines corresponding to nodes $i_j, j = 1, ..., l$, is changed from full waiting to partial waiting, the terms β_i in C_r corresponding to the machines with partial waiting satisfy the following: (i) the value of the terms $\beta_k, k \leq i_1, k \in C_r$, is at most B since the edge (s, i_1) exists in G^r (construction step 2); (ii) the value of the terms $\beta_k, k \in C_r, i_j < k \leq i_{j+1}, j = 1, ..., l-1$, is at most B since the edges $(i_j, i_{j+1}), j = 1, ..., l-1$,

exist in G^r (construction step 3); and (iii) the value of the terms $\beta_k, k > i_l, k \in C_r$, is at most B since the edge (i_l, t) exists in G^r (construction step 4). Thus, the explanations accompanying the construction steps show that if *each* graph $G^r, r = 1, ..., c$, has an s-t path, then changing the status of the machines corresponding to the nodes on these paths from full waiting to partial waiting provides us with a basic cycle π with a cycle time expression (3.4) in which all terms, except possibly the first term α, have value at most B. We record the precise statement below.

LEMMA 3.3 *Consider an initial partition $F = (V_1, V_2)$, where $V_2 = D_\delta$ and $V_1 = M \backslash V_2$. For $r = 1, ..., c$, let ρ^r be an s-t path in G^r and N^r be the set of nodes in ρ^r. The basic cycle π corresponding to $V_1^{'} = V_1 \backslash \{\cup_{r=1}^{c} N^r\}$ and $V_2^{'} = M \backslash V_1^{'}$ satisfies $\beta_k \leq B, \ \forall \ k \in V_2^{'}$.*

THEOREM 3.12 *Consider an initial partition $F = (V_1, V_2)$, where $V_2 = D_\delta$ and $V_1 = M \backslash V_2$. Given B, there exists a basic cycle π with $T(\pi) \leq B$ iff the sum of the shortest s-t paths in graphs $G^r, r = 1, ..., c$, is at most $B - \alpha$.*

Proof. For F, let V_1 and V_2 be the machines with full and partial waiting, respectively. Consider the shortest paths ρ^r in $G^r, r = 1, ..., c$, with respective lengths d^r such that $\sum_{r=1}^{c} d^r \leq B - \alpha$. Let N^r be the set of nodes in ρ^r. Consider the basic cycle $\bar{\pi}$ corresponding to $V_1^{'} = V_1 \backslash \{\cup_{r=1}^{c} N^r\}$ and $V_2^{'} = M \backslash V_1^{'}$. Note that $|V_2^{'}| \geq 2$. The cycle time for $\bar{\pi}$ can be written as $\max\{\alpha^{'}, \max_{i \in V_2^{'}} \beta_i^{'}\}$, where $\alpha^{'} = \alpha + \sum_{r=1}^{c} d^r$. Since $\sum_{r=1}^{c} d^r \leq B - \alpha$, we have $\alpha^{'} \leq B$. By Lemma 3.3, $\beta_i^{'} \leq B, \ \forall i \in V_2$. Also, as shown earlier, the new terms $\beta_i^{'}, i \in V_2^{'} \backslash V_2 = \cup_{r=1}^{c} N^r$, are less than the lower bound on the cycle time established in Theorem 3.2, and hence $\beta_i^{'} \leq B, \ \forall i \in V_2^{'} \backslash V_2$. Thus, $\beta_i^{'} \leq B, \ \forall i \in V_2^{'}$. Conversely, consider the unique basic cycle π corresponding to $(V_1^{'}, V_2^{'})$ with $T(\pi) \leq B$. We will construct the shortest paths ρ^r in $G^r, r = 1, .., c$, with respective lengths d^r such that $\sum_{r=1}^{c} d^r \leq B - \alpha$. In chain C_r, consider the nodes in $\xi^r = (V_2^{'} \backslash V_2) \cap C_r$. By construction, there exists an s-t path ρ^r in G^r with the nodes in ξ^r, in ascending order, as the intermediate nodes. Also, the sum of the lengths of ρ^r is at most $B - \alpha$; otherwise, by including the nodes in $\cup_r C_r$ in V_2, we would obtain a basic cycle corresponding to $(V_1^{'}, V_2^{'})$ with $T(\pi) > B$, which is a contradiction. ∎

EXAMPLE 3.3, CONTINUED: The shortest s-t path in G^2 is $s-7-11-t$ and has length $11 \leq B-\alpha = 14$. Thus, DQ has a positive answer. The new basic 1-unit cycle $\bar{\pi}$ corresponds to $V_1^{'} = V_1 \backslash \{7, 11\} = \{3, 9, 10, 13, 15\}$ and $V_2^{'} = M \backslash V_1^{'} = \{1, 2, 4, 5, 6, 7, 8, 11, 12, 14\}$. We get $\alpha^{'} = \alpha + 11 = 321, \beta_1^{'} = 310, \beta_2^{'} = 325, \beta_4^{'} = 330, \beta_5^{'} = 320, \beta_6^{'} = 333, \beta_7^{'} = 37, \beta_8^{'} = 319, \beta_{11}^{'} = 70, \beta_{12}^{'} = 314$, and $\beta_{14}^{'} = 312$. Observe that $\alpha^{'}, \beta_i^{'} \leq B = 334, \ \forall i \in V_2^{'}$. The cycle time $T(\bar{\pi}) = 333$. Continuing the binary search, it turns out that the 1-unit cycle $\bar{\pi}$, with $T(\bar{\pi}) = 333$, obtained above is in fact the optimal 1-unit cycle (i.e., $\pi^* = \bar{\pi}$). Thus, the optimal 1-unit cycle is

$$\pi^* = (A_0, A_{14}, A_{15}, A_{12}, A_{13}, A_{11}, A_8, A_9, A_{10}, A_7, A_6, A_5, A_4, A_2, A_3, A_1).$$

This completes the solution of the decision problem DQ. We now describe algorithm FindCycle that uses the solution of DQ within a standard binary search procedure.

Algorithm FindCycle

Input: $p_i, i = 1, ..., m; \delta; \epsilon; D_\delta = \{i \in M : p_i \geq \delta\}$; an Initial Partition $F = (V_1, V_2)$, where $V_2 = D_\delta$, $V_1 = M \backslash D_\delta$; the chains $C_r, r = 1, ..., c$, for F.

Step 1: Initialization: Set $UB = 2(m+1)\delta + 2(m+1)\epsilon$; $LB = 2(m+1)\delta - n_1\delta + \sum_{i \in V_1} p_i + 2(m+1)\epsilon$; $B = \lfloor \frac{LB+UB}{2} \rfloor$; $\mathcal{C} = \{C_r : r = 1, ..., c\}$; $\pi^* = \pi_D$.

Step 2: Construct and solve the shortest path problem for each $C_r \in \mathcal{C}$ to answer the decision question DQ for the value of B.

Step 3: If the answer to DQ is "yes", then update π^* to be the basic cycle $\bar{\pi}$ corresponding to $(V_1^{'}, V_2^{'})$ (as defined in the proof of Theorem 3.12) and set $UB = B$. Otherwise, let $LB = B$.

Step 4: If $UB - LB \geq 1$, then set $B = \lfloor \frac{UB+LB}{2} \rfloor$, and go to Step 2. Otherwise, terminate.

Note that in Step 1, UB is initialized to $2(m+1)\delta + 2(m+1)\epsilon$, which is the cycle time for the reverse cycle π_D under Case 4. Therefore, in Step 3 the answer to DQ is "yes" at least once. On termination, the basic cycle π^* is an optimum solution.

To compute the complexity of algorithm FindCycle, let $u_r = |V^r|$, $r = 1, ..., c$. The shortest path problem on $G^r(V^r, E^r)$ can be solved in time $O(u_r^2)$ via Dijkstra's algorithm [4]. Since $\sum_{r=1}^{c} u_r \leq m$, DQ can be solved in time $O(m^2)$. Since the cycle time of an optimal 1-unit cycle π^* satisfies $2(m+1)\delta - n_1\delta + \sum_{i \in M \setminus D_\delta} p_i + 2(m+1)\epsilon \leq T(\pi^*) \leq 2(m+1)\delta + 2(m+1)\epsilon$, algorithm FindCycle performs a binary search over an interval of length at most $m\delta$. Since all data are integral, the optimum cycle time is an integer and hence binary search requires time $O(log(m\delta))$. The running time of FindCycle is thus $O(m^2 log(m\delta))$. Note that we start with the same initial partition $F = (V_1, V_2), V_2 = D_\delta, V_1 = M \setminus V_2$, when solving each instance of problem DQ during the binary search.

REMARK 3.3 Consider a robotic cell in which the loaded travel time (δ_1) and the empty travel time (δ_2) are constant between any pair of machines with $\delta_1 > \delta_2$. It is easy to verify that the entire analysis can be extended to obtain a polynomial-time algorithm for obtaining an optimal 1-unit cycle in such robotic cells.

3.3.3 General Cases: Additive and Euclidean Travel-Time Cells

To solve the optimal 1-unit cycle problem in additive travel-time cells (problem $RF_m|(free,A,cyclic\text{-}1)|\mu$), Crama and van de Klundert [40] employ a concept that has been used to analyze the traveling salesman problem: the set of 1-unit pyramidal cycles [103].

DEFINITION 3.9 The 1-unit cycle $\pi = (A_0, A_{i_1}, A_{i_2}, \ldots, A_{i_m})$ is *pyramidal* if there exists a $k \in M$ such that $1 \leq i_1 < i_2 < \cdots < i_k = m$, and $m > i_{k+1} > i_{k+2} > \cdots > i_m \geq 1$. In such a cycle, $\mathcal{U} = \{i_1, i_2, \ldots, i_k\}$ is the set of *uphill activities* and $\mathcal{D} = \{i_{k+1}, i_{k+2}, \ldots, i_m\}$ is the set of *downhill activities*.

The permutations $\pi_U = (A_0, A_1, ..., A_m)$ and $\pi_D = (A_0, A_m, A_{m-1}, ..., A_1)$ are pyramidal, as is the permutation $(A_0, A_2, A_5, A_7, A_6, A_4, A_3, A_1)$ in a seven-machine cell. In an m-machine cell, there are 2^{m-1} pyramidal cycles. For the proofs of Theorems 3.13 and 3.14, we refer the reader to Crama and van de Klundert [40].

THEOREM 3.13 *The set of pyramidal 1-unit permutations is dominating among the class of 1-unit cycles.*

We can now prove Theorem 3.6 from Section 3.3.1.

Proof of Theorem 3.6. $T_a(\pi_D) = \max\{4m\delta+2(m+1)\epsilon,\ \max p_i+4(\delta+\epsilon)\}$. If $T_a(\pi_D) = \max p_i + 4(\delta+\epsilon)$, then it is optimal by Theorem 3.4. Assume $T_a(\pi_D) = 4m\delta+2(m+1)\epsilon$. The set of pyramidal cycles contains an optimal cycle [40]. Note that activity A_m is always considered to be an uphill activity, so π_D is the pyramidal cycle that corresponds to $U = \{m\}$.

Consider a general pyramidal cycle $\pi_p \neq \pi_D$, and let i, $1 \leq i \leq m-1$, be the smallest index of an uphill activity for cycle π_p. This implies that the form of cycle π_p is either

$$A_0A_iA_{i+1}\ldots A_m\ldots A_{i-1}A_{i-2}\ldots A_1 \quad \text{or}$$
$$A_0A_i\ldots A_m\ldots A_{i+1}A_{i-1}A_{i-2}\ldots A_1.$$

In either case, we can easily calculate lower bounds on the durations of the following nonoverlapping segments of the cycle:

1. From the start of activity A_i until the start of activity A_{i+1}: $\delta + 2\epsilon + p_{i+1}$
2. From the start of activity A_{i+1} until the start of activity A_{i-1}: $4\delta+2\epsilon$
3. From the start of activity A_{i-1} until the start of activity A_i: $\delta+2\epsilon+p_i$

Thus, we have the following lower bound for the cycle time:

$$T(\pi_p) \geq p_i + p_{i+1} + 6\delta + 6\epsilon \geq 4m\delta + 2(m+1)\epsilon = T(\pi_D)$$

The result now follows immediately. ∎

THEOREM 3.14 *For rational values of $p_i, i = 1, ..., m; \delta_i, i = 0, ..., m; \epsilon$, problem $RF_m|$(free,A,cyclic-1)$|\mu$ can be solved via a dynamic programming algorithm in time $O(m^3)$.*

For Euclidean travel-time cells, Brauner et al. [24] show that the decision problem corresponding to the optimum 1-unit cycle problem (i.e., problem $RF_m|$(free,E,cyclic-1)$|\mu$) is NP-complete.

For no-wait cells, Levner et al. [111] develop a polynomial-time algorithm for finding the minimum cycle time for Euclidean travel-time cells (problem $RF_m|$(no-wait,E,cyclic-1)$|\mu$). The algorithm, described later in Chapter 9, uses the processing times of the machines and the travel

times of the robot to derive infeasible intervals for the cycle time. The optimal cycle time is the smallest positive number not in these intervals. Obviously, this algorithm can be applied to the less general cases of additive (problem $RF_m|(\textit{no-wait,A,cyclic-1})|\mu$) and constant (problem $RF_m|(\textit{no-wait,C,cyclic-1})|\mu$) travel times, too.

For interval robotic cells with additive travel-times (problem $RF_m|(\textit{interval,A,cyclic-1})|\mu$), the robot move sequencing problem is NP-hard [41]. This implies that the problem is NP-hard for interval cells with Euclidean travel-times, too (problem $RF_m|(\textit{interval,E,cyclic-1})|\mu$). No results have been published for interval cells with constant travel time.

3.4 Calculation of Makespan of a Lot

A basic assumption in our analysis so far has been that the cell operates in steady state for an infinite amount of time. Since the amount of time required to reach steady state (starting with an empty cell) is finite (Brauner and Finke [22]), this transient phase has no contribution toward the long-term throughput of the cell. In practice, this is a reasonable argument for the production of high-demand items. In some situations, however, low demand or the complexities involved in implementing repetitive production results in small production lot-sizes. In such cases, a steady-state behavior is typically inappropriate; the makespan of the lot is a better measure of the production time. We now discuss three methods for calculating the makespan of a lot. One is graphical, the other two are algebraic.

3.4.1 A Graphical Approach

Herrmann et al. [83] use a directed acyclic graph to calculate a lot's makespan. Each node represents either the robot's movement or a machine's processing of a part. A node is labeled with the time it requires. Arcs indicate precedence constraints: $(j,k) \in A$ if action j must precede action k. For example, there is an arc originating at the node representing activity A_1 (labeled $\delta+2\epsilon$) whose destination is the node representing processing on M_2 (labeled p_2). In total, there are l nodes for each activity, where l is the lot size. The total time needed to process a lot equals

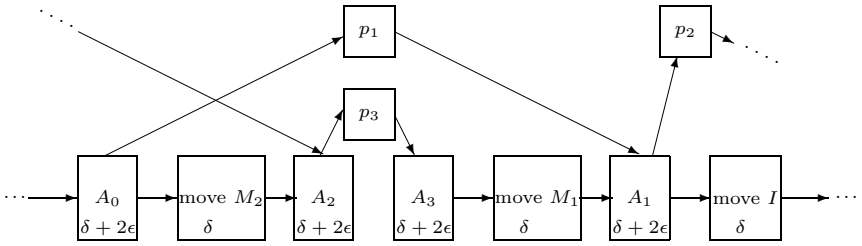


Figure 3.5. Portion of Graph to Calculate Makespan of a Lot Using Cycle (A_0, A_2, A_3, A_1).

the length of the network's longest path, called the *critical path*. This path is found via a dynamic programming algorithm. An example of a portion of such a graph for cycle (A_0, A_2, A_3, A_1) is in Figure 3.5.

This formulation is very useful for sensitivity analysis. The effect of an increase in the time of an operation (either a processing time or a movement time) depends on the size of the increase and whether that operation is on the critical path. If it is not on the critical path, an increase may have no effect on the makespan. However, a large enough increase may put the operation onto the critical path. Thus, the effect of changing one activity is a nondecreasing piecewise linear function.

3.4.2 Algebraic Approaches

Wood [159] calculates a semiconductor lot's total processing time by using two parameters. The first is the incremental lot cycle time t, which is defined as the average increase in the lot cycle time that results from a lot size increase of one wafer. The fixed time T represents the lot setup time that is independent of the lot size. If the lot size is l, then a lot's total processing time is $CT = T + lt$.

The cell's throughput is often improved by using multiple input devices, which are called *cassette ports* in this implementation. If there are n_L such ports and each holds a lot, then, by Little's [113] formula, the throughput is

$$\mu = \frac{n_L l}{T + lt}.$$

This assumes that the cell's performance is constrained by the input supply. If not, then $\mu = 1/t$ [159].

The previous two methods find a lot's total processing time by assuming that the cell is in steady state throughout its entire processing. We now examine a more realistic model by Perkinson et al. [128] that accounts for the times that the cell is not in steady state.

A lot's total processing includes five stages: loading the parts to be processed into the robotic cell (T_{load}), transition from start into steady state (T_A), steady state (T_S), processing the final parts (T_B), and unloading the completed parts (T_{unload}). Thus, the expression for the total lot makespan is $T_L = T_{load} + T_A + T_S + T_B + T_{unload}$. Here, T_{load} and T_{unload} are given constants. $T_S = FP(l - m + 1)$, where l is the lot size and FP is the fundamental period, i.e., the steady state per unit cycle time. The times for the transition periods are

$$\begin{aligned} T_A &= z(p + 2\delta) + \sum_{i=z+1}^{m-1} 2i\delta, \\ T_B &= z(p + 4\delta) + \sum_{i=z+1}^{m-1} 2(i+1)\delta - 3\delta, \end{aligned}$$

where p is the processing time for each machine, δ is the travel time between any two machines (note that $\epsilon = 0$ in this model), and $z = \min\{m-1, \lfloor((p/\delta)+2)/2\rfloor\}$ represents the maximum number of machines that can be in use before the processing speed becomes a bottleneck.

3.5 Quality of 1-Unit Cycles and Approximation Results

Having found optimal 1-unit cycles for problems $RF_m|$(*free,A,cyclic-1*)$|\mu$ and $RF_m|$(*free,C, cyclic-1*)$|\mu$, a question naturally arises: "Is an optimal 1-unit cycle superior to every non-trivial k-unit cycle, $k \geq 2$?" Sethi et al. [142] prove this to be true for $RF_2|$(*free,A,cyclic-1*)$|\mu$ and conjectured it to be so for $m \geq 3$. The attraction of this possibility is obvious: 1-unit cycles are the easiest to understand, analyze, and control. If they also have the highest throughput, there is no reason to consider the more complex and more numerous multi-unit cycles.

For $RF_3|$(*free,A,cyclic-1*)$|\mu$, Crama and van de Klundert [42] and Brauner and Finke [20] each prove that the conjecture is true. The con-

jecture also holds for $RF_3|(free,C,cyclic\text{-}1)|\mu$. However, the conjecture does not hold for $RF_4|(free,A,cyclic\text{-}1)|\mu$; Brauner and Finke [18, 22] provide a counterexample. For $RF_4|(free,C,cyclic\text{-}1)|\mu$, consider the cell with the following data [45]: $p_1 = 22, p_2 = 1, p_3 = 1, p_4 = 22, \delta = 4, \epsilon = 0$. The best 1-unit cycle is (A_0,A_4,A_3,A_1,A_2); the cycle time is 39. The best 2-unit cycle is $(A_0,A_4,A_3,A_1,A_0,A_4,A_2,A_3,A_1,A_2)$, whose per unit cycle time is 38. Note that although this 2-unit cycle dominates all 1-unit cycles and all other 2-unit cycles in this cell, we cannot assert its optimality over all k-unit cycles, $k \geq 1$. By Theorem 3.2, the lower bound on the optimal value is 34, so there may be a k-unit cycle, $k \geq 3$, that has per unit cycle time less than 38.

Similar results for $RF_m|(no\text{-}wait,A,cyclic\text{-}1)|\mu$ and $RF_m|(interval,A,cyclic\text{-}1)|\mu$ are summarized in Crama et al. [39].

Even though 1-unit cycles do not dominate, their simplicity still makes them attractive in practice. We have seen that the reverse cycle π_D is optimal under certain conditions. Crama and van de Klundert [40] show that π_D is a 2-approximation for $RF_m|(free,A,cyclic\text{-}1)|\mu$ (i.e., the cycle time of π_D is at most twice that of an optimal 1-unit cycle). Brauner and Finke [21] show that if the optimum per unit cycle time over all k-unit cycles is T_{opt}, then

$$T_a(\pi_D) \leq \left(2 - \frac{\delta_1 + \delta_{m+1}}{\delta_1 + \delta_{m+1} + \sum_{i=2}^{m} \delta_i}\right) T_{opt} \leq 2T_{opt}.$$

For $RF_m|(free,C,cyclic\text{-}1)|\mu$, we have

$$T_c(\pi_D) \leq \left(\frac{2(m+1)(\delta+\epsilon)}{(m+2)\delta + 2(m+1)\epsilon}\right) T_{opt} \leq \left(\frac{2(m+1)\delta}{(m+2)\delta}\right) T_{opt} \leq 2T_{opt}.$$

3.5.1 Additive Travel-Time Cells

In this section, we develop a 1.5-approximation algorithm for the optimal per unit cycle time for additive travel-time cells. We begin with some elementary results, discuss a dominant subclass of 1-unit cycles, present the algorithm, and prove that it provides the stated bound. We start by establishing a lower bound for the per unit cycle time. The following result extends Theorem 3.2, and is based on results by Crama and van de Klundert [40].

THEOREM 3.15 *For any k-unit ($k \geq 1$) cycle π in an additive travel-time cell, the per unit cycle time $T_a(\pi)/k$ satisfies*

$$\frac{T_a(\pi)}{k} \geq \max\left\{2(m+1)(\epsilon+\delta) + \sum_{i=1}^{m} \min\{p_i, \delta\}, \max_{1\leq i\leq m} p_i + 4\delta + 4\epsilon\right\}$$

Proof. We begin by addressing the first argument. Each unit produced requires that $m+1$ activities be performed. Each activity $A_i, i = 0, \ldots, m$, includes unloading machine M_i and loading machine M_{i+1}, for a total of $2(m+1)\epsilon$ per part. Each activity A_i also includes a loaded forward robot movement that requires δ time, for a total of $(m+1)\delta$. Furthermore, because the robot's final location M_0 is the same as its initial location, each forward movement across the interval (M_i, M_{i+1}) must have a later corresponding backward movement across (M_i, M_{i+1}). This accounts for an additional $(m+1)\delta$.

The summation term in the first argument represents the time between activities. After the robot completes activity A_i, $i = 0, \ldots, m-1$, by loading machine M_{i+1}, it either waits at machine M_{i+1} for the duration of its processing (p_{i+1}), or it moves to another machine (δ, at minimum) to begin another activity. The minimum movement after activity A_m is counted in the first term.

The second argument represents the minimum time between successive loadings of a given machine M_i: processing (p_i), activity A_i ($\delta + 2\epsilon$), move from M_{i+1} to M_{i-1} (2δ), and activity A_{i-1} ($\delta + 2\epsilon$). ∎

Recall that $D_\delta = \{i : p_i \geq \delta\}$, $D_\delta^c = M \setminus D_\delta$, and $|D_\delta|$ is the cardinality of D_δ. For convenience, we denote the per unit cycle time of an optimal k-unit cycle ($k \geq 1$) in an additive travel-time cell by Ω_a. Using this notation, Theorem 3.15 can be restated as

$$\Omega_a \geq \max\{2(m+1)\epsilon + [2(m+1) + |D_\delta|]\delta + \sum_{i\in D_\delta^c} p_i, \max_{1\leq i\leq m} p_i + 4\delta + 4\epsilon\}. \tag{3.11}$$

The following two results are based on Sethi et al. [142], Crama and van de Klundert [40], and Dawande et al. [46].

LEMMA 3.4 *If $\max_{1\leq i\leq m} p_i + 4\delta + 4\epsilon \geq 4m\delta + 2(m+1)\epsilon$, then π_D is an optimal k-unit cycle, $k \geq 1$.*

LEMMA 3.5 *If* $p_i + p_{i+1} \geq (4m-6)\delta + 2(m-2)\epsilon, i = 1, ..., m-1$, *then* π_D *is an optimal* k*-unit cycle,* $k \geq 1$.

3.5.1.1 Pyramidal Cycles

As described in Section 3.3.3, a pyramidal cycle partitions M into two sets: the indices of uphill activities denoted by $\mathcal{U} = \{0, i_1, i_2, \ldots, i_k = m\}$ and the indices of downhill activities denoted by $\mathcal{D} = \{i_{k+1}, i_{k+2}, \ldots, i_m\}$. π_U and π_D are pyramidal, as is $(A_0, A_2, A_5, A_7, A_6, A_4, A_3, A_1)$. Given the dominance of pyramidal cycles (Theorem 3.13), it is only natural that they be considered when seeking a cycle that provides an efficient bound for the optimum per unit cycle time.

An expression for the cycle time of a pyramidal cycle π_p can be derived as follows. Each of the $m+1$ activities has one unloading and one loading, so $T_a^l = 2(m+1)\epsilon$. During the uphill portion, the robot performs activities $A_0, A_{i_1}, A_{i_2}, \ldots, A_m$, $0 < i_1 < i_2 < \cdots < m$. Hence, the robot travels once the path from M_0 to M_{m+1} requiring time $(m+1)\delta$, no matter how many uphill activities there are. During the downhill portion, the robot travels from M_{m+1} to M_0, which requires a minimum time of $(m+1)\delta$. For each activity A_i during the downhill portion, the robot travels from M_i to M_{i+1}, then from M_{i+1} to M_i, before continuing to M_0. Hence, each downhill activity adds 2δ to the cycle's movement time. Thus, the total time for robot moves is $T_a^m = 2(m+1+|\mathcal{D}|)\delta$.

The robot will have full waiting at M_i if and only if $i-1 \in \mathcal{U}$ and $i \in \mathcal{U}$. We designate the set of indices of such machines by $\mathcal{U}' = \{i : i-1 \in \mathcal{U}, i \in \mathcal{U}\}$. Partial waiting can occur at all other machines. Therefore,

$$T_a(\pi_p) = 2(m+1)\epsilon + 2(m+1+|\mathcal{D}|)\delta + \sum_{i \in \mathcal{U}'} p_i + \sum_{i \in M \setminus \mathcal{U}'} w_i. \quad (3.12)$$

3.5.1.2 A 1.5-Approximation Algorithm

We now show that repeating an optimal 1-unit cycle k times is a 1.5-approximation of an optimal k-unit cycle. We do this by developing an $O(n)$ algorithm that finds a 1-unit cycle that can be shown to provide a 1.5-approximation. In this algorithm, the first three steps check for cases in which an optimal cycle is known. Step 4 considers the case in which, by inequality (3.11), π_D provides a bound of 1.5. For the remaining case,

we construct a special pyramidal cycle that has a total partial waiting time of zero.

Algorithm ACell

Input: The data for an additive travel-time simple robotic cell: m, δ, ϵ, $p_i, i = 1, \ldots, m$.

Step 1: If $p_i \leq \delta$, $\forall i$, then output $\pi_U = (A_0, A_1, \ldots, A_m)$. Stop.

Step 2: If $\max_{1 \leq i \leq m} p_i + 4\delta + 4\epsilon \geq 4m\delta + 2(m+1)\epsilon$, then output $\pi_D = (A_0, A_m, A_{m-1}, \ldots, A_2, A_1)$. Stop.

Step 3: If $p_i + p_{i+1} \geq (4m-6)\delta + 2(m-2)\epsilon$, $i = 1, \ldots, m-1$, then output π_D. Stop.

Step 4: If $|D_\delta| \geq \frac{2}{3}m$, then output π_D. Stop.

Step 5: If $|D_\delta| < \frac{2}{3}m$, then partition the indices $1, 2, \ldots, m$ into two sets, $\mathcal{U}$ and $\mathcal{D}$, as follows:

a) Place m into $\mathcal{U}$.

b) If $p_m \geq \delta$ or $p_{m-1} \geq \delta$, then place $m-1$ in $\mathcal{D}$. Otherwise, place $m-1$ in $\mathcal{U}$.

c) For $i = m-2, m-3, \ldots, 2, 1$:

If $p_i \geq \delta$, then place i in $\mathcal{D}$.

Else if $i+1 \in \mathcal{D}$ and placing i into $\mathcal{U}$ may cause M_{i+1} to have positive partial waiting, i.e., if

$$p_{i+1} > 2(m-i)\delta + 2(m-i-1)\epsilon + \sum_{\substack{j=i+2 \\ j \in \mathcal{U}'}}^{m} p_j + \sum_{\substack{j=i+2 \\ j \in \mathcal{D}}}^{m-1} 2\delta, \tag{3.13}$$

then place i in $\mathcal{D}$.

Otherwise, place i in $\mathcal{U}$.

Loop

Step 6: Form pyramidal cycle π_p by making the activities corresponding to the elements of $\mathcal{U}$ uphill, and those corresponding to the elements of $\mathcal{D}$ downhill. Output π_p. Stop.

Steps 1, 2, 3, and 4 compare $p_i, i = 1, \ldots, m$, or $p_i + p_{i+1}, i = 1, \ldots, m-1$, to a constant, so each requires time $O(m)$. Step 5 compares p_i to a constant and p_{i+1} to a sum, so it requires time $O(m)$. Step 6 orders the m activities according to the algorithm for forming pyramidal cycles, so it requires time $O(m)$. Therefore, algorithm ACell requires time $O(m)$.

EXAMPLE 3.4 We illustrate Step 5. $m = 10,\ \delta = 2,\ \epsilon = 1$. The vector of processing times is $p = (1, 20, 1, 1, 50, 1, 1, 1, 12, 8)$. Observe that

$$\begin{aligned}
p_{10} > \delta &\Rightarrow A_9 \in \mathcal{D} \\
p_9 = 12 > 4\delta + 2\epsilon &\Rightarrow A_8 \in \mathcal{D} \\
p_7 < \delta, p_8 < 8\delta + 4\epsilon &\Rightarrow A_7 \in \mathcal{U} \\
p_6 < \delta, p_7 < 12\delta + 6\epsilon &\Rightarrow A_6 \in \mathcal{U} \\
p_5 > \delta &\Rightarrow A_5 \in \mathcal{D} \\
p_5 = 50 > 16\delta + 10\epsilon + p_7 &\Rightarrow A_4 \in \mathcal{D} \\
p_3 < \delta, p_4 < 18\delta + 10\epsilon + p_7 &\Rightarrow A_3 \in \mathcal{U} \\
p_2 > \delta &\Rightarrow A_2 \in \mathcal{D} \\
p_1 < \delta, p_2 = 20 < 24\delta + 16\epsilon + p_7 &\Rightarrow A_1 \in \mathcal{U}
\end{aligned}$$

Therefore, $\pi_p = (A_0, A_1, A_3, A_6, A_7, A_{10}, A_9, A_8, A_5, A_4, A_2)$ and its cycle time is $T_a(\pi_p) = 22\epsilon + 2(11 + 5)\delta + p_1 + p_7 \ = \ 88$.

THEOREM 3.16 *Algorithm ACell is a 1.5-approximation algorithm for the optimum per unit cycle time, and this bound is tight. Consequently, we have a 1.5-approximation for an optimal multi-unit cyclic solution.*

Proof.

a) From inequality (3.11), if $p_i \leq \delta,\ \forall i$, then $\Omega_a \geq 2(m + 1)(\delta + \epsilon) + \sum_{i=1}^{m} p_i = T_a(\pi_U)$. Therefore, π_U is optimal.

b) If $\max_{1 \leq i \leq m} p_i + 4\delta + 4\epsilon \geq 4m\delta + 2(m + 1)\epsilon$, then by Lemma 3.4, π_D is optimal.

c) If $p_i + p_{i+1} \geq (4m - 6)\delta + 2(m - 2)\epsilon,\ i = 1, \ldots, m - 1$, then by Lemma 3.5, π_D is optimal.

d) We can now assume that $\max_{1\le i\le m} p_i + 4\delta + 4\epsilon < 4m\delta + 2(m+1)\epsilon$. If $|D_\delta| \ge \frac{2}{3}m$, then by inequality (3.11),

$$\begin{aligned}\Omega_a &\ge \left[2(m+1) + \frac{2}{3}m\right]\delta + 2(m+1)\epsilon + \sum_{i\in D_\delta^c} p_i \\ &\ge \frac{8}{3}m\delta + 2(m+1)\epsilon.\end{aligned}$$

Hence,

$$\begin{aligned}T_a(\pi_D) = 4m\delta + 2(m+1)\epsilon &\le \frac{4m\delta + 2(m+1)\epsilon}{\frac{8}{3}m\delta + 2(m+1)\epsilon}\Omega_a \\ &\le 1.5\Omega_a. \qquad (3.14)\end{aligned}$$

e) For $|D_\delta| < \frac{2}{3}m$, we first derive an expression for $T_a(\pi_p)$.

Claim: Either pyramidal cycle π_p is optimal, or it has no partial waiting: $\sum_{i\in M\setminus\mathcal{U}'} w_i = 0$.

Proof of Claim.

i) If $i \in \mathcal{D}$ and $i-1 \in \mathcal{D}$, then $w_i > 0$ implies that $T_a(\pi_p) = p_i + 4\delta + 4\epsilon$, in which case π_p is optimal by Theorem 3.15.

ii) By construction, $i \in \mathcal{U}$ implies that either $p_i \le \delta$ or $i = m$. Hence, if $i \in \mathcal{U}$ and $i-1 \in \mathcal{D}$, then either $w_i = 0$ or $w_m > 0$. If $w_m > 0$, then $T_a(\pi_p) = p_m + 4\delta + 4\epsilon$, so π_p is optimal by Theorem 3.15.

iii) If $i \in \mathcal{U}$ and $i-1 \in \mathcal{U}$, then they are consecutive activities, so M_i has full waiting equal to p_i.

iv) Consider now the case in which $i \in \mathcal{U}$ and $i+1 \in \mathcal{D}$. By the three previous cases, for the largest $i+1$ in this case, there is no positive partial waiting between the loading of M_{i+1} and when the robot returns to unload it. We first prove that $w_{i+1} > 0$ only if inequality (3.13) is true, which contradicts membership in this case. We do this by showing that the right-hand side of inequality (3.13) represents the time between the loading of M_{i+1} and when the robot returns to unload it in this case.

After M_{i+1} is loaded by A_i, the robot completes the uphill portion of the cycle by traveling to M_{m+1}, performing other uphill

activities along the way. This movement time is $(m-i)\delta$. It then begins the downhill portion by traveling from M_{m+1} to M_{i+1}, for a movement time of $(m-i)\delta$. In addition, each activity performed during the downhill portion causes the robot to travel an additional 2δ (this amount is represented in the fourth term). Because π_p is pyramidal, between A_i and A_{i+1}, the robot performs activities $A_{i+2}, A_{i+3}, \ldots, A_{m-1}, A_m$, though not necessarily in this order, since, most likely, some will be uphill and some will be downhill. Each activity requires one loading and one unloading: $2(m-i-1)\epsilon$. During the uphill portion of this sequence, the robot will have full waiting at M_j if and only if $j \in \mathcal{U}'$ (third term). Therefore, $w_{i+1} > 0$ only if inequality (3.13) is true.

Hence, by construction, for the next such machine, say $M_{i'+1}$, there is no positive partial waiting between the loading of $M_{i'+1}$ and the time that the robot returns to unload it. Thus, if $p_{i'+1}$ does not satisfy inequality (3.13), $w_{i'+1} = 0$. It is easy to see that this argument can be repeatedly applied to the remaining machines M_{j+1} for which $j \in \mathcal{U}, j+1 \in \mathcal{D}$. □

Thus, from equation (3.12),

$$T_a(\pi_p) = 2(m+1+|\mathcal{D}|)\delta + 2(m+1)\epsilon + \sum_{i\in\mathcal{U}'} p_i.$$

Note that $T_a(\pi_p) \le T_a(\pi_D)$. First observe that if $\mathcal{D} = D_\delta$, then

$$\begin{aligned} T_a(\pi_p) &\le \frac{2(m+1+|D_\delta|)\delta + 2(m+1)\epsilon + \sum_{i\in\mathcal{U}'} p_i}{[2(m+1)+|D_\delta|]\delta + 2(m+1)\epsilon + \sum_{i\in D_\delta^c} p_i}\Omega_a \\ &\le \frac{2(m+1+|D_\delta|)\delta}{[2(m+1)+|D_\delta|]\delta}\Omega_a \le 1.25\Omega_a, \end{aligned}$$

since this expression is increasing in $|D_\delta|$ and $|D_\delta| < \frac{2}{3}m$.

If $\mathcal{D}\backslash D_\delta \neq \emptyset$, then let i' be the smallest element of $\mathcal{D}\backslash D_\delta$. This implies that $i'+1$ satisfies inequality (3.13). Since each element of $\mathcal{D}\backslash D_\delta$ has a corresponding element of $D_\delta \subset \mathcal{D}$ that satisfies inequality (3.13) (except for $m-1$ if $p_{m-1} < \delta$ and $p_m > \delta$), the number of elements in $\mathcal{D}$ that are greater than $i'+1$ is at least $2(|\mathcal{D}\backslash D_\delta|-1)-1$. Hence, $m \notin \mathcal{D}$ implies that $i'+1 \le m - 2(|\mathcal{D}\backslash D_\delta|-1) - 2$, so $i' \le m -$

$2(|\mathcal{D}\backslash D_\delta| - 1) - 3 = m - 2|\mathcal{D}\backslash D_\delta| - 1$. Therefore, inequality (3.13) becomes

$$\begin{aligned} p_{i'+1} &> 2(m-i')\delta + 2(m-i'-1)\epsilon + \sum_{\substack{j=i+2 \\ j\in\mathcal{D}}}^{m-1} 2\delta + \sum_{\substack{j=i+2 \\ j\in\mathcal{U}'}}^{m} p_j \\ &\geq (4|\mathcal{D}\backslash D_\delta| + 2)\delta + 4|\mathcal{D}\backslash D_\delta|\epsilon + 4(|\mathcal{D}\backslash D_\delta| - 1)\delta \\ &\quad + \sum_{\substack{j=i+2 \\ j\in\mathcal{U}'}}^{m} p_j. \end{aligned}$$

Recall that $\Omega_a \geq p_{i'+1} + 4\delta + 4\epsilon$. Thus,

$$\begin{aligned} \Omega_a &\geq 8(|\mathcal{D}\backslash D_\delta| + 2)\delta + 4(|\mathcal{D}\backslash D_\delta| + 1)\epsilon + \sum_{\substack{j=i+2 \\ j\in\mathcal{U}'}}^{m} p_j \\ &\geq 8(|\mathcal{D}\backslash D_\delta| + 2)\delta + 4(|\mathcal{D}\backslash D_\delta| + 1)\epsilon. \end{aligned}$$

If $|\mathcal{D}\backslash D_\delta| \geq \frac{m}{3}$, then $\Omega_a \geq (\frac{8}{3}m + 2)\delta + \left(\frac{4}{3}m + 4\right)\epsilon$, so

$$T_a(\pi_p) \leq T_a(\pi_D) \leq \frac{4m\delta + 2(m+1)\epsilon}{\frac{8}{3}m\delta + (\frac{4}{3}m + 2)\epsilon}\Omega_a \leq 1.5\Omega_a.$$

If $|\mathcal{D}\backslash D_\delta| < \frac{m}{3}$, then

$$T_a(\pi_p) \leq \frac{2(m + 1 + \frac{m}{3} + |D_\delta|)\delta}{2(m+1)\delta + |D_\delta|\delta}\Omega_a.$$

This expression is increasing in $|D_\delta|$ and $|D_\delta| < \frac{2}{3}m$, so

$$T_a(\pi_p) \leq \frac{4m\delta}{\frac{8}{3}m\delta}\Omega_a = 1.5\Omega_a.$$

By combining these results with inequality (3.14), we see that algorithm ACell provides the proposed bound.

We now show that the algorithm provides an asymptotically tight bound. Let $m = 3k, k \in \mathbb{Z}^+$, $\epsilon = 0$, $p_{3j} = 0$, and $p_{3j-1} = p_{3j-2} = 2\delta, j = 1, 2, \ldots, k$. Obviously, $|D_\delta| = 2k$. Define the pyramidal cycle π_1 by $\mathcal{U} = \{3j, 3j-1 : j = 1, \ldots, k\}$ and $\mathcal{D} = \{3j-2 : j = 1, \ldots, k\}$. Then $T_a(\pi_1) = 2(3k+1)\delta + 2k\delta = (8k+2)\delta = \Omega_a$, so π_1 is optimal. Since

$|D_\delta| \geq \frac{2}{3}m$, algorithm ACell will yield π_D as the heuristic solution in Step 3, and its cycle time is

$$T_a(\pi_D) = 4m\delta + 2(m+1)\epsilon = 4(3k)\delta \leq \frac{12k}{8k+2}\Omega_a \rightarrow 1.5\Omega_a \text{ as } k \rightarrow \infty.$$

■

COROLLARY 3.2 *In an additive travel-time cell, a k-unit cyclic solution obtained by repeating an optimal 1-unit cyclic solution k times is a 1.5-approximation to an optimal k-unit cyclic solution, $k \geq 1$.*

REMARK 3.4 For the more general additive travel-time case in which $d(M_i, M_{i+1}) = \delta_i$, $i = 0, \ldots, m$, Crama and van de Klundert [40] show that π_D provides a 2-approximation to the optimum per unit cycle time.

For additive travel-time cells, Geismar et al. [62] improve on the result of algorithm ACell by providing a polynomial-time cyclic solution that is a 10/7-approximation to an optimum cyclic solution. We discuss this algorithm next.

3.5.1.3 A 10/7-Approximation for Additive Cells

We start by establishing another lower bound for the per unit cycle time. Consider the set $D_2 = \{i : p_i \geq 2\delta\}$. Thus, $D_2^c = \{i : p_i < 2\delta\}$. Obviously, $D_2 \subseteq D_\delta$. We define a *run* of length ℓ to be any maximal sequence of ℓ consecutive indices that belong to D_2, i.e., $(j, j+1, j+2, \ldots, j+\ell-1)$ is a run with length ℓ if $\{j, j+1, j+2, \ldots, j+\ell-1\} \subseteq D_2$, and $\{j-1, j+\ell\} \cap D_2 = \emptyset$. The number of runs of length ℓ in D_2 is denoted r_ℓ, so $|D_2| = \sum_{\ell=1}^{m} \ell r_\ell$, and the total number of runs is $\sum_{\ell=1}^{m} r_\ell$.

In the following theorem, the amount that $i \in D_2$ contributes to a lower bound on the optimum per unit cycle time depends on the size ℓ of the run to which i belongs. This lower bound includes $2(m+1)(\delta+\epsilon)$ plus the following: if a run has length ℓ and ℓ is even, then the run's elements require a total extra time of $\ell\delta$. If ℓ is odd, then the run's elements require a total extra time of $(\ell+1)\delta$. Before we state and prove the theorem, the examples below provide motivation for the result.

EXAMPLE 3.5 Consider a four-machine cell with $p_1 = p_3 = 2\delta$, $p_2 = p_4 = 0$, $\epsilon = 0$. According to Theorem 3.2, $\Omega_a \geq 12\delta$. However, Theorem 3.17 states that $\Omega_a \geq 14\delta$. There are several 1-unit cycles that

achieve this latter optimum value: $(A_0, A_2, A_4, A_3, A_1)$, $(A_0, A_1, A_2, A_3, A_4)$, $(A_0, A_1, A_2, A_4, A_3)$, $(A_0, A_2, A_3, A_4, A_1)$.

EXAMPLE 3.6 For the five-machine cell with $p_2 = p_3 = p_4 = 2\delta$, $p_1 = p_5 = 0$, $\epsilon = 0$, we have $\Omega_a \geq 15\delta$ from Theorem 3.2, and $\Omega_a \geq 16\delta$ from Theorem 3.17. Again, several 1-unit cycles achieve this latter optimum value: $(A_0, A_1, A_3, A_5, A_4, A_2)$, $(A_0, A_1, A_4, A_5, A_3, A_2)$.

THEOREM 3.17 *For any k-unit ($k \geq 1$) cycle π in an additive travel-time cell, the per unit cycle time $T(\pi)/k$ satisfies*

$$\frac{T(\pi)}{k} \quad \geq \quad 2(m+1)(\delta+\epsilon) + 2\sum_{\ell=1}^{m} \left\lfloor \frac{\ell+1}{2} \right\rfloor r_\ell \delta. \tag{3.15}$$

Proof. Each unit produced requires that $m+1$ activities be performed. Each activity $A_i, i = 0, \ldots, m$, includes unloading machine M_i and loading machine M_{i+1}, for a total of $2(m+1)\epsilon$ per part. Each activity A_i also includes a loaded forward robot movement from M_i to M_{i+1} that requires δ time, for a total of $(m+1)\delta$. Furthermore, because the robot's final location M_0 is the same as its initial location, each forward movement across the interval (M_i, M_{i+1}) must have a later corresponding backward movement across (M_i, M_{i+1}). This accounts for an additional $(m+1)\delta$.

We now justify the second term of inequality (3.15). For ℓ even, since $D_2 \subseteq D_\delta$, Theorem 3.2 implies that a run of ℓ machines in D_2 adds $2\lfloor(\ell+1)/2\rfloor\delta = \ell\delta$ to the minimum per unit cycle time. For ℓ odd, we first consider a run of length 1, then extend the result to larger runs. Let $i-1 \in D_2^c$, $i \in D_2$, $i+1 \in D_2^c$, where $D_2^c = M \backslash D_2$. We have the following cases for activity subsequences within a k-unit cycle:

1) For any occurrence of the subsequence (A_{i-1}, A_i), the robot has full waiting at machine M_i for duration $p_i \geq 2\delta$.

2) For any occurrence of the subsequence $(A_0, A_{j_1}, A_{j_2}, \ldots, A_{j_q}, A_i)$ within a k-unit cycle, where $j_s \geq i+1$ for some $s \in \{1, \ldots, q\}$ and $j_u \neq i$, $\forall u \in \{1, \ldots, q\}$, the robot twice crosses the interval (M_i, M_{i+1}), once forward and once reverse, *before* performing A_i. This adds 2δ to the cycle time in addition to that added by the performance of activity A_i and its corresponding backward movement across the interval (M_i, M_{i+1}).

3) For any occurrence of the subsequence $(A_0, A_{j_1}, A_{j_2}, \ldots, A_{j_q}, A_i)$, $j_s \in \{0, 1, \ldots, i-1\}, \forall s \in \{1, \ldots, q\}$ and $j_q \neq i-1$, the robot crosses the interval (M_{i-1}, M_i) without a part immediately before executing activity A_i, which requires additional time δ. Before the next instance of A_i, the robot must perform A_{i-1}, so it must return to M_{i-1}. This requires crossing (M_{i-1}, M_i) in the reverse direction, which requires at least one more extra δ.

Therefore, a run of length 1 adds an extra 2δ to the minimum per unit cycle time.

Notice that the argument that assigns an extra 2δ to an element i of a one-machine run uses robot travel either over the interval (M_{i-1}, M_i) or (M_i, M_{i+1}). Thus, it cannot be applied to consecutive machines of a multi-machine run. However, the argument can be applied to n consecutive one-machine runs. Consider the following sequence of machines: $2i \in D_2$, $i = a, a+1, \ldots, a+n-1$; $2i-1 \in D_2^c$, $i = a, a+1, \ldots, a+n$. See Figure 3.6.

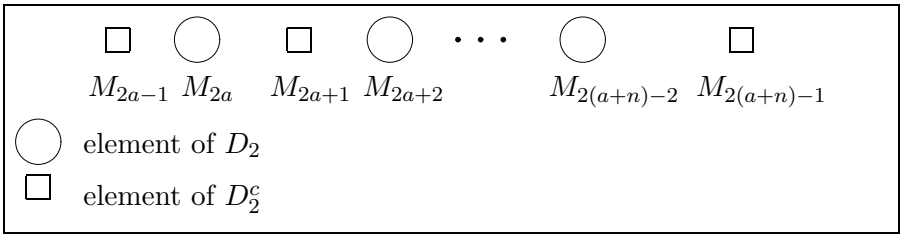


Figure 3.6. n Consecutive One-Machine Runs.

Here, the machines $M_{2a}, M_{2(a+1)} \ldots, M_{2(a+n-1)}$ form n consecutive one-machine runs, which together add $2n\delta$ to the minimum per unit cycle time. Because increasing a machine's processing time cannot reduce the cycle time for a given cycle, a run of $2n-1$ machines must add as much time to the minimum per unit cycle time as do n consecutive one-machine runs. Therefore, a run of $2n-1$ machines adds at least $2n\delta$ to the minimum per unit cycle time. If we let $\ell = 2n-1$, it follows that a run of ℓ machines in D_2 adds $2\lfloor(\ell+1)/2\rfloor\delta$ to the minimum per unit cycle time. ■

Algorithm ACellnew provides a cyclic solution whose per unit cycle time is within a factor of 10/7 of the optimum per unit cycle time. Its

goal is to create a pyramidal cycle with a maximal number of uphill activities, no full waiting greater than or equal to 2δ, and no positive partial waiting. The first two steps check for cases in which an optimal cycle is known. Afterward, Steps 3, 4, and 5 each loop through the indices of M. Step 3 ensures that there is no full waiting greater than or equal to 2δ: if $p_i \geq 2\delta$, then at least one of i and $i-1$ must be in $\mathcal{D}$. Step 4 moves any element of $\mathcal{U}$ whose corresponding machine could have positive partial waiting into $\mathcal{D}$. For $i \in \mathcal{D}$ and $i-1 \in \mathcal{U}$, if M_i could have positive partial waiting, then Step 5 moves $i-1$ to $\mathcal{D}$.

Algorithm ACellnew

Input: The data for an additive travel-time simple cell: $m,\ \delta,\ \epsilon, p_i, i = 1, \ldots, m$.

Step 1: If $p_i \leq \delta,\ \forall i$, then output $\pi_U = (A_0, A_1, \ldots, A_m)$. Stop.

Step 2: If $\max_{1 \leq i \leq m} p_i + 4\delta + 4\epsilon \geq 4m\delta + 2(m+1)\epsilon$, or if $p_i + p_{i+1} \geq (4m-6)\delta + 2(m-2)\epsilon,\ i = 1, \ldots, m-1$, then output $\pi_D = (A_0, A_m, A_{m-1}, \ldots, A_2, A_1)$. Stop.

Step 3: Divide the indices $1, 2, \ldots, m$ into two sets $\mathcal{U}$ and $\mathcal{D}$ such that the resulting pyramidal cycle π_p has no machine with full waiting greater than or equal to 2δ and a maximal number of indices are in $\mathcal{U}$.

a) $\mathcal{U} = \{0, m\}$, $\mathcal{D} = \emptyset$.

b) For $i = 1, \ldots, m-1$:

If ($p_i \geq 2\delta$ and $i - 1 \in \mathcal{U}$), then place i into $\mathcal{D}$.

Else place i into $\mathcal{U}$.

Loop

c) If ($m - 1 \in \mathcal{U}$ and $p_m \geq 2\delta$), then place $m-1$ into $\mathcal{D}$.

Step 4: Ensure that no machine corresponding to an element of $\mathcal{U}$ has positive partial waiting.

For $i = 2, \ldots, m-1$:

If $i \in \mathcal{U}$ and $i-1 \in \mathcal{D}$, and leaving i in $\mathcal{U}$ may cause M_i to have positive partial waiting, i.e., if

$$p_i > 2i\delta + 2(i-1)\epsilon + \sum_{\substack{j=1 \\ j \in \mathcal{U}'}}^{i-2} p_j + \sum_{\substack{j=1 \\ j \in \mathcal{D}}}^{i-2} 2\delta, \tag{3.16}$$

then place i into $\mathcal{D}$.

Loop

Step 5: Improve the partition to avoid positive partial waiting at the machines in $\mathcal{D}$ (i.e., $w_i = 0, \forall i \in \mathcal{D}$).

For $i = m-2, m-3, \ldots, 1$:

If $i \in \mathcal{U}$ and $i+1 \in \mathcal{D}$ and

$$p_{i+1} > 2(m-i)\delta + 2(m-i-1)\epsilon + \\ + \sum_{\substack{j=i+2 \\ j \in \mathcal{U}'}}^{m} p_j + \sum_{\substack{j=i+2 \\ j \in \mathcal{D}}}^{m-1} 2\delta, \tag{3.17}$$

then place i into $\mathcal{D}$.

Loop

Step 6: Form pyramidal cycle π_p by making the activities corresponding to the elements of $\mathcal{U}$ uphill, and those corresponding to the elements of $\mathcal{D}$ downhill. Output π_p. Stop.

Steps 1 and 2 compare $p_i, i = 1, \ldots, m$, to a constant, so each requires time $O(m)$. Steps 3, 4, and 5 each loop once through the m machines. In each loop, p_i is compared to a constant or a sum, which requires time $O(m)$. Step 6 orders the m activities according to the algorithm for forming pyramidal cycles, so it requires time $O(m)$. Therefore, algorithm ACellnew requires time $O(m)$.

EXAMPLE 3.7 We illustrate Steps 3, 4, and 5. $m = 14$, $\delta = 2$, $\epsilon = 1$. The vector of processing times is

$$p = (5, 7, 10, 30, 15, 3, 12, 20, 1, 38, 2, 18, 1, 7).$$

Step 3 operates as follows:

$$\begin{aligned}
p_1 \geq 2\delta \Rightarrow 1 \in \mathcal{D} \qquad & p_8 \geq 2\delta, 7 \in \mathcal{D} \Rightarrow 8 \in \mathcal{U}, \\
p_2 \geq 2\delta, 1 \in \mathcal{D} \Rightarrow 2 \in \mathcal{U} \qquad & p_9 < 2\delta \Rightarrow 9 \in \mathcal{U}, \\
p_3 \geq 2\delta, 2 \in \mathcal{U} \Rightarrow 3 \in \mathcal{D} \qquad & p_{10} \geq 2\delta, 9 \in \mathcal{U} \Rightarrow 10 \in \mathcal{D}, \\
p_4 \geq 2\delta, 3 \in \mathcal{D} \Rightarrow 4 \in \mathcal{U} \qquad & p_{11} < 2\delta \Rightarrow 11 \in \mathcal{U}, \\
p_5 \geq 2\delta, 4 \in \mathcal{U} \Rightarrow 5 \in \mathcal{D} \qquad & p_{12} \geq 2\delta, 11 \in \mathcal{U} \Rightarrow 12 \in \mathcal{D}, \\
p_6 < 2\delta \Rightarrow 6 \in \mathcal{U} \qquad & p_m = p_{14} \geq 2\delta \Rightarrow m - 1 = 13 \in \mathcal{D}, \\
p_7 \geq 2\delta, 6 \in \mathcal{U} \Rightarrow 7 \in \mathcal{D} \qquad & 14 = m \Rightarrow 14 \in \mathcal{U},
\end{aligned}$$

so after Step 3,

$$\begin{aligned}
\mathcal{U} &= \{2, 4, 6, 8, 9, 11, 14\}, \\
\mathcal{D} &= \{1, 3, 5, 7, 10, 12, 13\}.
\end{aligned}$$

For Step 4, we examine only indices in $\mathcal{U}$, other than $m = 14$, by using inequality (3.16):

$p_2 = 7 \leq 4\delta + 2\epsilon = 10$, so 2 stays in $\mathcal{U}$.

$p_4 = 30 > 8\delta + 6\epsilon + 2\delta = 26$, so 4 moves to $\mathcal{D}$.

$p_6 = 3 \leq 12\delta + 10\epsilon + 6\delta = 46$, so 6 stays in $\mathcal{U}$.

Since the test value is increasing as the indices increase and each of p_8, p_9, and p_{11} is less than 46, it follows that each of the indices 8, 9, and 11 stays in $\mathcal{U}$. After Step 4,

$$\begin{aligned}
\mathcal{U} &= \{2, 6, 8, 9, 11, 14\}, \\
\mathcal{D} &= \{1, 3, 4, 5, 7, 10, 12, 13\}.
\end{aligned}$$

For Step 5, we use inequality (3.17) to examine only indices in $\mathcal{D}$ whose predecessors are in $\mathcal{U}$:

$p_3 = 10 \leq 24\delta + 22\epsilon + p_9 + 12\delta = 95$, so 2 stays in $\mathcal{U}$,

$p_7 = 12 \leq 16\delta + 14\epsilon + p_9 + 6\delta = 59$, so 6 stays in $\mathcal{U}$,

$p_{10} = 38 > 10\delta + 8\epsilon + 4\delta = 36$, so 9 moves to $\mathcal{D}$,

$p_{12} = 18 > 6\delta + 4\epsilon + 2\delta = 20$, so 11 moves to $\mathcal{D}$.

After Step 5,

$$\begin{aligned}
\mathcal{U} &= \{2, 6, 8, 14\}, \\
\mathcal{D} &= \{1, 3, 4, 5, 7, 9, 10, 11, 12, 13\}.
\end{aligned}$$

Therefore,

$$\pi_p = (A_0, A_2, A_6, A_8, A_{14}, A_{13}, A_{12}, A_{11}, A_{10}, A_9, A_7, A_5, A_4, A_3, A_1),$$

and $T(\pi_p) = 30\epsilon + 2(15 + 10)\delta = 130$.

THEOREM 3.18 *Algorithm ACellnew is a* $10/7$*-approximation algorithm for the optimum per unit cycle time, and this bound is tight. Consequently, we have a* $10/7$*-approximation for an optimal multi-unit cyclic solution.*

Proof. From inequality (3.11), if $p_i \leq \delta,\ \forall i$, then $\Omega_a \geq 2(m+1)(\delta + \epsilon) + \sum_{i=1}^{m} p_i = T(\pi_U)$. Therefore, π_U is optimal. If $\max_{1\leq i\leq m} p_i + 4\delta + 4\epsilon \geq 4m\delta + 2(m+1)\epsilon$, then by Corollary 3.4, π_D is optimal. If $p_i + p_{i+1} \geq (4m-6)\delta + 2(m-2)\epsilon,\ i = 1, \ldots, m-1$, then π_D is optimal by Theorem 3.6.

The strategy of the remainder of the proof is summarized in four parts as follows:

1. Show that π_p has no positive partial waiting, unless it is optimal.
2. Find an instance for which $T(\pi_p)/\Omega_a$ is maximal.
3. Prove that $T(\pi_p)/\Omega_a \leq 10/7$ for this instance.
4. Show that the bound is tight.

Part 1 of the proof is established by the following claim:
Claim: Either pyramidal cycle π_p is optimal, or it has no partial waiting: $\sum_{i\in M\setminus\mathcal{U}'} w_i = 0$.
Proof of Claim.

i) If $i \in \mathcal{U}$ and $i-1 \in \mathcal{U}$, then they are consecutive activities, so M_i has full waiting equal to p_i.

ii) Suppose $i \in \mathcal{U}$ and $i-1 \in \mathcal{D}$. After machine M_i is loaded during activity A_{i-1}, the robot travels to M_0 and then back to M_i, which requires $2i\delta$ time. During the downhill portion of this trip, the robot performs $|\{j \in \mathcal{D} : 1 \leq j \leq i-2\}|$ activities, each of which requires 2δ movement time. If an uphill activity A_j has full waiting, then $j \in \mathcal{U}'$, and the robot waits for time p_j. Because π_p is pyramidal, between machine M_i's loading and unloading, $i-1$ activities

$(A_0, A_1, \ldots, A_{i-2})$ are performed, each requiring 2ϵ for loading and unloading. Therefore, to have positive partial waiting at M_i, p_i must satisfy inequality (3.16), in which case Step 4 of ACellnew moves i to $\mathcal{D}$.

iii) Consider now the case in which $i \in \mathcal{U}, i+1 \in \mathcal{D}$. We first prove that $w_{i+1} > 0$ only if inequality (3.17) is true. We do this by showing that the right-hand side of inequality (3.17) represents the time between the loading of M_{i+1} and when the robot returns to unload it in this case.

After M_{i+1} is loaded by A_i, the robot completes the uphill portion of the cycle by traveling to M_{m+1}, performing other uphill activities along the way. This movement time is $(m-i)\delta$. It then begins the downhill portion by traveling from M_{m+1} to M_{i+1}, for a movement time of $(m-i)\delta$. In addition, each activity performed during the downhill portion causes the robot to travel for an additional time of 2δ (this amount is represented in the fourth term of inequality (3.17)). Because π_p is pyramidal, between A_i and A_{i+1}, the robot performs activities $A_{i+2}, A_{i+3}, \ldots, A_{m-1}, A_m$, though not necessarily in this order, since, most likely, some will be uphill and some will be downhill. Each activity requires one loading and one unloading: $2(m-i-1)\epsilon$. During the uphill portion of this sequence, the robot will have full waiting at M_j if and only if $j \in \mathcal{U}'$ (third term). Thus, if $p_{i'+1}$ does not satisfy inequality (3.17), $w_{i'+1} = 0$.

iv) Let $\mathcal{D}' = \{i : i \in \mathcal{D}, i-1 \in \mathcal{D} \cup \{0\}\}$. We show that if $w_i > 0$ for some $i \in \mathcal{D}'$, then $T(\pi_p) = \max_{i \in \mathcal{D}'} p_i + 4\delta + 4\epsilon$. If $w_i > 0$, then the time between the completion of machine M_i's loading and the beginning of its unloading is p_i. The time between the beginning of its unloading and the completion of its loading is calculated as follows: unload M_i (ϵ), carry part to M_{i+1} (δ), load M_{i+1} (ϵ), travel to M_{i-1} (2δ), wait (if necessary) before unloading M_{i-1} (w_{i-1}), unload M_{i-1} (ϵ), carry part to M_i (δ), and load M_i (ϵ). Therefore, $T(\pi_p) = p_i + 4\delta + 4\epsilon + w_{i-1}$, $\forall i \in \mathcal{D}'$. Note that $w_{i-1} > 0$ only if $i-1 \in \mathcal{D}'$ (by the previous three cases) and $p_{i-1} > p_i$. Hence, if $p_{i'} = \max_{i \in \mathcal{D}'} p_i$, then $T(\pi_p) = p_{i'} + 4\delta + 4\epsilon$. It follows that if $w_i > 0$ for some $i \in \mathcal{D}'$, then $T(\pi_p) = \max_{i \in \mathcal{D}'} p_i + 4\delta + 4\epsilon$, and π_p is optimal by Theorem 3.2.

This completes the proof of our claim. □

From equality (3.12),

$$T(\pi_p) = 2(m+1+|\mathcal{D}|)\delta + 2(m+1)\epsilon + \sum_{i\in\mathcal{U}'} p_i.$$

We now perform Part 2 of the proof by determining a case for which $T(\pi_p)/\Omega_a$ is maximal. Because $i \in \mathcal{U}' \Rightarrow p_i < 2\delta$, $T(\pi_p)$ is maximized by maximizing $\mathcal{D}$. To maximize the size of $\mathcal{D}$, first assume that $\forall i \in D_2$ that was not placed into $\mathcal{D}$ by Step 3, p_i is large enough so that i is placed into $\mathcal{D}$ by Step 4. In addition, the processing time of the first element of each run of elements of D_2 must be large enough to satisfy inequality (3.17). This will cause its predecessor (an element of D_2^c) to be placed into $\mathcal{D}$ by Step 5 of ACellnew. Therefore, $|\mathcal{D}| \le |D_2| + \sum_{\ell=1}^{m} r_\ell = \sum_{\ell=1}^{m} (\ell+1) r_\ell$. Hence, by using the bound for Ω_a in Theorem 3.17, we have

$$\begin{aligned} T(\pi_p) &= 2(m+1+|\mathcal{D}|)\delta + 2(m+1)\epsilon + \sum_{i\in\mathcal{U}'} p_i \qquad (3.18)\\ &\le 2\left(m+1+\sum_{\ell=1}^{m}(\ell+1)r_\ell\right)\delta + 2(m+1)\epsilon + \sum_{i\in\mathcal{U}'} p_i \\ &\le \frac{2\left(m+1+\sum_{\ell=1}^{m}(\ell+1)r_\ell\right)\delta + 2(m+1)\epsilon + \sum_{i\in\mathcal{U}'} p_i}{2(m+1)(\delta+\epsilon) + 2\sum_{\ell=1}^{m}\left\lfloor\frac{\ell+1}{2}\right\rfloor r_\ell\delta}\Omega_a \\ &= \frac{A+2(m+1)\epsilon}{B+2(m+1)\epsilon}\Omega_a, \end{aligned}$$

where $A = 2\left(m+1+\sum_{\ell=1}^{m}(\ell+1)r_\ell\right)\delta + \sum_{i\in\mathcal{U}'} p_i$ and $B = 2(m+1)\delta + 2\sum_{\ell=1}^{m}\lfloor(\ell+1)/2\rfloor r_\ell\delta$. Since $r_\ell \ge 0, \ell = 1,...,m$; $p_i \ge 0, i = 1,...,m$; $\epsilon \ge 0$; and $\delta \ge 0$, it follows that $B \le A$. Therefore,

$$\begin{aligned} T(\pi_p) &\le \frac{A+2(m+1)\epsilon}{B+2(m+1)\epsilon}\Omega_a \\ &\le \frac{A}{B}\Omega_a \\ &= \frac{2\left(m+1+\sum_{\ell=1}^{m}(\ell+1)r_\ell\right)\delta + \sum_{i\in\mathcal{U}'} p_i}{2(m+1)\delta + 2\sum_{\ell=1}^{m}\left\lfloor\frac{\ell+1}{2}\right\rfloor r_\ell\delta}\Omega_a. \end{aligned}$$

The ratio is maximized by maximizing $\sum_{\ell=1}^{m}(\ell+1)r_\ell / \sum_{\ell=1}^{m}\lfloor(\ell+1)/2\rfloor r_\ell$. (Note that if we try to maximize by maximizing $|\mathcal{U}'|$, then (3.18) implies that $T(\pi_p) \le (4m+2)\delta + 2(m+1)\epsilon$, and Theorem 3.2 implies

$\Omega_a \geq (3m+2)\delta + 2(m+1)\epsilon$. Thus, $T(\pi_p) \leq (4/3)\Omega_a)$. Therefore, each run must be small. Because the lower bounds generated by odd-sized runs differ from those generated by even-sized runs, we allow for both: $r_1 \geq 0$, $r_2 \geq 0$, and $r_\ell = 0$, $\ell \geq 3$. Hence,

$$T(\pi_p) \quad \leq \quad \frac{2(m+1)\delta + 2(2r_1+3r_2)\delta + \sum_{i\in\mathcal{U}'} p_i}{2(m+1)\delta + 2(r_1+r_2)\delta}\Omega_a.$$

Since $2r_1 + 3r_2 \leq m+1$ and $p_i < 2\delta \ \forall i \in \mathcal{U}'$, $T(\pi_p)/\Omega_a$ is maximized by setting $2r_1 + 3r_2 = m+1$ and minimizing the denominator, so choose $r_1 = 0$ and $r_2 = (m+1)/3$, which implies that $|D_2| = (2/3)(m+1)$ ($D_2 = \{1, 2, 4, 5, 7, 8, \ldots, 3k-2, 3k-1\}$, $k \in \mathbb{Z}^+$, where $m = 3k-1$).

Before we proceed with the remainder of the proof, we provide an example to illustrate the subsequent analysis. Let $m = 8$ and the vector of processing times be $p = (3\delta, 5\delta + 2\epsilon, 0, 17\delta + 8\epsilon, 15\delta + 8\epsilon, 0, 5\delta + 2\epsilon, 0)$. After Step 3, $\mathcal{D} = \{1, 4, 7\}$. Step 4 places indices 2 and 5 into $\mathcal{D}$. Indices 3 and 6 are placed into $\mathcal{D}$ by Step 5. Hence, $\pi_p = (A_0, A_8, A_7, A_6, A_5, A_4, A_3, A_2, A_1)$, and $T(\pi_p) = 32\delta + 18\epsilon$. Using the lower bounds from Theorem 3.2 and Theorem 3.17, we have $\Omega_a \geq \max\{23\delta + 18\epsilon, 24\delta + 18\epsilon\} = 24\delta + 18\epsilon$. Thus, for this example, the worst-case bound is $T(\pi_p)/\Omega_a \leq 4/3$.

Having found set D_2, we find an instance for which $|\mathcal{D}|$ is maximal. Let $m = 3k-1$, $p_{3j-2} \geq 2\delta$, $p_{3j-1} \geq 2\delta$, $j = 1, \ldots, k$, and $p_{3j} < 2\delta$, $j = 1, \ldots, k-1$. Specific values for p_{3j-2}, p_{3j-1}, and p_{3j} will be determined later.

Recall that from inequality (3.11), $\Omega_a \geq 2(m+1)(\delta+\epsilon) + |D_\delta|\delta + \sum_{i\in D_\delta^c} p_i$. For the current instance, $D_\delta = \{3j-2, 3j-1 : j = 1, \ldots, k\} \cup \{3j : \delta \leq p_{3j} < 2\delta\}$. So by Theorem 3.2,

$$\Omega_a \geq 8k\delta + 6k\epsilon + |\{3j : \delta \leq p_{3j} < 2\delta\}|\delta + \sum_{3j\in D_\delta^c} p_{3j}, \tag{3.19}$$

which implies that for maximizing $T(\pi_p)/\Omega_a$, we must have $p_i = 0$, $i \in D_2^c$. This is easy to see. Recall that $D_2^c = \{i : p_i < 2\delta\}$. Since we want both the third and fourth terms of (3.19) to be zero, there should be (i) no j such that $\delta \leq p_{3j} < 2\delta$, and (ii) no j such that $0 < p_{3j} < \delta$. Thus, imposing $p_i = 0$, $i \in D_2^c$, the lower bound of inequality (3.19) becomes $\Omega_a \geq 8k\delta + 6k\epsilon$.

Because either π_p is optimal or $\max\{T(\pi_p)\} = 4m\delta + 2(m+1)\epsilon$, and $\Omega_a \geq \max p_i + 4\delta + 4\epsilon$, the theorem holds for all cases in which

$4m\delta + 2(m+1)\epsilon \leq (10/7)(\max p_i + 4\delta + 4\epsilon)$. Therefore, for all i, we can assume that

$$\begin{aligned} p_i &< \frac{7}{10}(4m\delta + 2(m+1)\epsilon) - 4\delta - 4\epsilon \\ &\leq (8.4k - 6.8)\delta + (4.2k - 4)\epsilon, \end{aligned} \tag{3.20}$$

using $m = 3k - 1$.

Step 3 places each index $3j-2$ in $\mathcal{D}$. For any index $3j-1, j = 1, \ldots, k-1$, that is placed in $\mathcal{D}$ by Step 4, $T(\pi_p)$ increases by $2\delta - p_{3j} = 2\delta > 0$. Hence, it appears that to maximize $T(\pi_p)/\Omega_a$, $p_{3j-1}, j = 1, \ldots, k-1$, should be large enough to satisfy inequality (3.16). Note that increasing the values of $p_{3j-1}, j = 1, \ldots, k-1$, has no effect on Ω_a.

We now show that not all indices $3j-1, j = 1, ..., k-1$, can be placed in $\mathcal{D}$ without violating inequality (3.20). From inequality (3.16),

$$p_{3j-1} > 2(3j-1)\delta + 2(3j-2)\epsilon + \sum_{\substack{i=1 \\ i \in \mathcal{D}}}^{3j-3} 2\delta,$$

because $p_i = 0, i \in \mathcal{U}'$.

Consider the summation term. If starting at $j = 1$, each $3j-1$ is placed in $\mathcal{D}$, then for a given j^*, we have $|\{i \in \mathcal{D} : i \leq 3j^* - 3\}| = |\{3j-1, 3j-2 : j \leq j^*-1\}| = 2j^*-2$. This would mean that $3j^*-1 \in \mathcal{D}$ if

$$\begin{aligned} p_{3j^*-1} &> 2(3j^* - 1)\delta + 2(3j^* - 2)\epsilon + 2(2j^* - 2)\delta \\ &= (10j^* - 6)\delta + (6j^* - 4)\epsilon. \end{aligned} \tag{3.21}$$

However, combining this with inequality (3.20) implies

$$\begin{aligned} (10j^* - 6)\delta + (6j^* - 4)\epsilon &< (8.4k - 6.8)\delta + (4.2k - 4)\epsilon, \text{ so} \\ j^* &< \frac{(8.4k - .8)\delta + 4.2k\epsilon}{10\delta + 6\epsilon} < 0.84k. \end{aligned}$$

Thus, after Step 4, $T(\pi_p)/\Omega_a$ is maximized by $\mathcal{D} = \{3j - 2 : 1 \leq j \leq k\} \cup \{3j - 1 : 1 \leq j < .84k\}$.

Step 5 loops backward through $\mathcal{D}$, which is increased by large values of p_{3j-2}. Specifically, by inequality (3.17), $3j - 3$, $j = 2, \ldots, k$, is added

to $\mathcal{D}$ if

$$p_{3j-2} \;>\; 2(3k-3j+2)\delta + 2(3k-3j+1)\epsilon + \sum_{\substack{i=3j-1 \\ i\in\mathcal{D}}}^{3k-2} 2\delta.$$

To simplify the summation term of this inequality, observe that

$$\begin{aligned} |\{i\in\mathcal{D} : 3j-1 \le i \le 3k-2\}| &= \lfloor 1.84k\rfloor - 2j + 1, \qquad 1 \le j < 0.84k \\ |\{i\in\mathcal{D} : 3j-1 \le i \le 3k-2\}| &= k-j+1, \qquad j \ge 0.84k. \end{aligned}$$

Hence, for $j \ge 0.84k$, we must have $p_{3j-2} > (8k-8j+6)\delta+(6k-6j+2)\epsilon$. For $j < 0.84k$, by combining inequalities (3.17) and (3.20), we get that index $3j-3$ is moved to $\mathcal{D}$ only if $p_{3j-2} > (9.68k-10j+6)\delta+(6k-6j+2)\epsilon$ and

$$\begin{aligned} (9.68k-10j+6)\delta + (6k-6j+2)\epsilon &< (8.4k-6.8)\delta + (4.2k-4)\epsilon, \text{ so} \\ j &> \frac{(1.28k+12.8)\delta + (1.8k+6)\epsilon}{10\delta+6\epsilon} \\ &\ge 0.128k. \end{aligned}$$

It follows that our approximating cycle π_p is the pyramidal cycle formed by $\mathcal{D} = \{3j-2 : 1 \le j \le k\} \cup \{3j-1 : 1 \le j < .84k\} \cup \{3j : 0.128k < j \le k-1\}$, and $|\mathcal{D}| = \lfloor 2.712k\rfloor$.

Having specified an instance for which $T(\pi_p)/\Omega_a$ is maximal, we now show that this ratio is at most 10/7 (Part 3 of the proof):

$$\begin{aligned} \frac{T(\pi_p)}{\Omega_a} &\le \frac{2(m+1)(\delta+\epsilon)+2|\mathcal{D}|\delta}{2(m+1)(\delta+\epsilon)+|D_\delta|\delta} \\ &\le \frac{2(m+1)+2|\mathcal{D}|}{2(m+1)+|D_\delta|} \le \frac{6k+2(2.712)k}{8k} \approx \frac{10}{7}. \end{aligned}$$

We now prove the tightness of the 10/7 bound (Part 4). Let $m = 3k-1, \epsilon = 0$, $p_{3j} = 0, 1 \le j \le k$. $p_{3j-2} = 2\delta$, $1 \le j \le 0.128k$; $p_{3j-2} = (9.68k-10j^*+6)\delta+1$, $0.128 < j \le k$; $p_{3j-1} = (10j-6)\delta+1$, $1 \le j < 0.84k$; $p_{3j-1} = 2\delta$, $0.84k \le j \le k$. An optimal 1-unit pyramidal cycle π_3 is defined by $\mathcal{D} = \{3j-2 : j = 1,\ldots,k\}$ and $\mathcal{U} = \{3j-1, 3j : j = 1,\ldots,k\}$. In π_3, the partial waiting times are $w_2 = 1, w_{3k-2} = 1$, and $w_i = 0$ otherwise. Therefore, $T(\pi_3) = 2[(3k-1)+1]\delta + 2k\delta + 2 = 8k\delta + 2 \to \Omega_a$ as $k \to \infty$.

We have seen that algorithm ACellnew defines π_p by $\mathcal{D} = \{3j-2 : 1 \le j \le k\} \cup \{3j-1 : 1 \le j < 0.84k\} \cup \{3j : j = 0.128 < j \le k\}$, and that

$$T(\pi_p) = 11.424k\delta \le \frac{11.424k\delta}{8k\delta}\Omega_a \approx \frac{10}{7}\Omega_a.$$

■

Note that algorithm ACellnew repeats a specific 1-unit cycle to prove the 10/7 guarantee; this 1-unit cycle may not necessarily be optimal. If, instead, we repeat an *optimal* 1-unit cycle, then it follows that the result will be at least as good. Recall that an optimal 1-unit cycle in additive travel-time cells can be obtained in polynomial time (Crama and van de Klundert [40]). The following result is, therefore, immediate.

COROLLARY 3.3 *In an additive travel-time cell, a k-unit cyclic solution obtained by repeating an optimal 1-unit cyclic solution k times is a polynomial-time* 10/7*-approximation to an optimal k-unit cyclic solution,* $k \ge 1$.

We can use results from this proof to determine optimal cycles for certain cases that are common in practice. Define the *alternating pyramidal cycle* by $\pi_a = (A_0, A_2, A_4, \ldots, A_{m-2}, A_m, A_{m-1}, A_{m-3}, \ldots, A_3, A_1)$ if m is even and $\pi_a = (A_0, A_2, A_4, \ldots, A_{m-3}, A_m, A_{m-1}, A_{m-2}, A_{m-4}, \ldots, A_3, A_1)$ if m is odd. From equality (3.12),

$$T(\pi_a) = \begin{cases} (3m+2)\delta + 2(m+1)\epsilon + \sum_{i=1}^{m} w_i, & \text{for } m \text{ even}, \\ \\ (3m+3)\delta + 2(m+1)\epsilon + \sum_{i=1}^{m} w_i, & \text{for } m \text{ odd}. \end{cases}$$

If $w_m > 0$ and $w_{m-1} = 0$, then $T(\pi_a) = p_m + 4\delta + 4\epsilon$, so π_a is optimal. If m is odd, $w_{m-1} > 0$, and $w_{m-2} = 0$, then $T(\pi_a) = p_{m-1} + 4\delta + 4\epsilon$, so π_a is optimal. Recall that $\sum_{i=1}^{m-1} w_i = 0$ only if p_i does not satisfy inequality (3.16), $\forall i \in \mathcal{U}$, and p_i does not satisfy inequality (3.17), $\forall i \in \mathcal{D}$, as we saw in the proof of Theorem 3.18. This leads to the following corollary.

COROLLARY 3.4 *Given an additive travel-time robotic cell with* $p_i \ge 2\delta$, $\forall i$. *The alternating cycle* π_a *is optimal if* $p_i \le (3i-2)\delta + 2(i-1)\epsilon$, *for all even* i, $i \le m-2$, *and either of the following conditions hold:*

1. m *is even and* $p_i \le (3m-3i+1)\delta + 2(m-i)\epsilon$, *for* i *odd; or*

2. *m is odd and $p_i \leq (3m - 3i + 2)\delta + 2(m - i)\epsilon$, for i odd, $i \neq m$.*

Proof. The corollary's conditions imply that π_a can have positive partial waiting only at M_m, or at M_{m-1} for m odd. In either case, π_a is optimal as shown above. Otherwise, the corollary's conditions imply that π_a has no positive partial waiting, so

$$T(\pi_a) = \begin{cases} (3m+2)\delta + 2(m+1)\epsilon, & \text{for } m \text{ even}, \\ \\ (3m+3)\delta + 2(m+1)\epsilon, & \text{for } m \text{ odd}. \end{cases} \tag{3.22}$$

By Theorem 3.17,

$$\begin{aligned} \frac{T(\pi)}{k} &\geq 2(m+1)(\delta+\epsilon) + 2\left\lfloor \frac{m+1}{2} \right\rfloor \delta \\ &= \begin{cases} (3m+2)\delta + 2(m+1)\epsilon, & \text{for } m \text{ even}, \\ \\ (3m+3)\delta + 2(m+1)\epsilon, & \text{for } m \text{ odd}. \end{cases} \end{aligned}$$

Therefore, π_a is optimal. ∎

COROLLARY 3.5 *Given an additive travel-time robotic cell with $p_i \geq \delta$, $\forall i$ and $p_i \leq (3i-2)\delta + 2(i-1)\epsilon$, for all even i, $i \leq m-2$. If m is even and $p_i \leq (3m-3i+1)\delta + 2(m-i)\epsilon$, for i odd, then the alternating cycle π_a is optimal. If m is odd and $p_i \leq (3m-3i+2)\delta + 2(m-i)\epsilon$, for i odd, $i \neq m$, then $T(\pi_a) - \Omega_a \leq \delta$, so π_a is asymptotically optimal as m increases to infinity.*

Proof. The corollary's conditions imply that π_a can have positive partial waiting only at M_m, or at M_{m-1} for m odd. In either case, π_a is optimal as shown above. By Theorem 3.2, $\Omega_a \geq (3m+2)\delta + 2(m+1)\epsilon$. Hence, by equality (3.22), π_a is optimal if m is even. For m odd, we have $T(\pi_a) - \Omega_a \leq \delta$. Therefore, $T(\pi_a)/\Omega_a \leq 1 + (\delta/\Omega_a) \to 1$ as m increases to infinity. So, π_a is asymptotically optimal. ∎

3.5.2 Constant Travel-Time Cells

In this section, we develop a 1.5-approximation algorithm for the optimum per unit cycle time for constant travel-time cells. The following lower bound for the per unit cycle time is based on results from Dawande et al. [47].

THEOREM 3.19 *For any k-unit cycle π, $k \geq 1$, in a constant travel-time cell, the per unit cycle time $T_c(\pi)/k$ satisfies*

$$\frac{T_c(\pi)}{k} \geq \max\{2(m+1)\epsilon + \sum_{i=1}^{m} \min\{p_i, \delta\} + (m+2)\delta, \max_{1\leq i\leq m} p_i + 3\delta + 4\epsilon\}.$$

Proof. We show that $T_c(\pi) \geq 2k(m+1)\epsilon + k\sum_{i=1}^{m} \min\{p_i, \delta\} + k(m+2)\delta$, and that $T_c(\pi) \geq k(\max_{1\leq i\leq m} p_i + 3\delta + 4\epsilon)$. The result follows immediately.

Consider the first argument. A k-unit cycle consists of $k(m+1)$ activities. Each activity requires one loading and one unloading, so the total time for these actions is $2k(m+1)\epsilon$. Before each activity A_i, $i = 1, \ldots, m$, time will be taken either by a robot move (δ) or a processing time (p_i). This time is represented by the second term. The robot never has to wait for processing to complete before executing activity A_0, so the total time taken before all k A_0's is $k\delta$; this is included in the last term. The last term also includes the robot's movement time while performing the $k(m+1)$ activities (transfer of a part from M_i to M_{i+1}, $i = 0, \ldots, m$), which is $k(m+1)\delta$.

For the second argument, observe that the sequence of actions between the start of M_i's unloading and the completion of its next loading must include activity A_i, travel to M_{i-1}, and activity A_{i-1}. At minimum, this time is $3\delta + 4\epsilon$, since each activity requires $\delta + 2\epsilon$ time. Thus, the minimum time between each loading of M_i is $p_i + 3\delta + 4\epsilon$. In a k-unit cycle, this must be done k times. ∎

Let Ω_c denote the per unit cycle time of an optimal k-unit cycle, $k \geq 1$, in a constant travel-time robotic cell. Recall the definition for the following set: $D_\delta = \{i : p_i \geq \delta\}$. Using these definitions, Theorem 3.19 can be restated as

$$\Omega_c \geq \max\left\{2(m+1)\epsilon + \sum_{i\in D_\delta^c} p_i + [m+2+|D_\delta|]\delta, \max_{1\leq i\leq m} p_i + 3\delta + 4\epsilon\right\}. \quad (3.23)$$

The following result is from Dawande et al. [47].

LEMMA 3.6 *If* $\max_{1\le i\le m} p_i + 3\delta + 4\epsilon \ge 2(m+1)(\delta+\epsilon)$, *then* π_D *is optimal.*

Recall from Section 3.3.2 that, for a 1-unit basic cycle π_B, the cycle time is $T_c(\pi_B) = \max\{\alpha, \beta_i | i \in V_2\}$, where

$$\begin{aligned} \alpha &= 2(m+1)\delta + 2(m+1)\epsilon + \sum_{i\in V_1}(p_i - \delta), \\ \beta_i &= p_i + 3\delta + 4\epsilon + (r_i + q_i)(\delta + 2\epsilon) + \sum_{j\in X_i \cup Y_i} p_j. \end{aligned}$$

For a constant travel-time cell, the *initial partition* (V_1, V_2) divides the set of machine indices M into those that represent machines with full waiting and those that represent machines with partial waiting by the following assignments: $V_1 = D_\delta^c$ and $V_2 = D_\delta$. Note that for the initial partition,

$$\alpha = [m + 2 + |D_\delta|]\delta + 2(m+1)\epsilon + \sum_{i\in D_\delta^c} p_i. \tag{3.24}$$

So, by inequality (3.23),

$$\Omega_c \ge \max\{\alpha, \max_{1\le i\le m} p_i + 3\delta + 4\epsilon\}. \tag{3.25}$$

3.5.2.1 A 1.5-Approximation Algorithm

We now show that repeating an optimal 1-unit cycle k times is a 1.5-approximation of an optimal k-unit cycle. We do this by developing an $O(m)$ algorithm that finds a 1-unit cycle that can be shown to provide a 1.5-approximation. In the algorithm, the first three steps check for cases in which an optimal cycle is known. Step 4 considers the case in which, by inequality (3.23), π_D provides a 1.5-bound. For the remaining case, we construct a special basic cycle π'_B in which V'_2 includes D_δ plus some other machine indices. The cycle π'_B satisfies the following properties: (i) for all $i \in D_\delta \subset V'_2$, $X_i = Y_i = \emptyset$, so $\beta'_i = p_i + 3\delta + 4\epsilon$, and (ii) $\alpha' > \beta'_i, i \in V'_2 \backslash D_\delta$. Therefore, either $T_c(\pi'_B) = \alpha'$, or π'_B is optimal.

Algorithm CCell

Input: The data for a constant travel-time simple robotic cell: m, δ, ϵ, $p_i, i = 1, \ldots, m$.

Step 1: If $p_i \leq \delta$, $\forall i$, then output $\pi_U = (A_0, A_1, \ldots, A_m)$. Stop.

Step 2: If $\max_{1\leq i\leq m} p_i + 3\delta + 4\epsilon \geq 2(m+1)(\delta+\epsilon)$, then output $\pi_D = (A_0, A_m, A_{m-1}, \ldots, A_2, A_1)$. Stop.

Step 3: If $p_i \geq \delta$, $\forall i$, then output π_D. Stop.

Step 4: If $|D_\delta| \geq \frac{1}{3}(m-2)$, then output π_D. Stop.

Step 5: If $|D_\delta| < \frac{1}{3}(m-2)$, then form the set V_2' as follows:

$$\begin{aligned} V_2' &= D_\delta \cup \{j : j \in D_\delta^c,\ j = i+1,\ i \in D_\delta\} \cup \\ &\quad \{j : j \in D_\delta^c,\ j = i-1,\ i \in D_\delta\}. \end{aligned}$$

Step 6: Form a basic cycle π_B' such that the machines corresponding to the elements of V_2' have partial waiting and those corresponding to the elements of $V_1' = M \backslash V_2'$ have full waiting. Output π_B'. Stop.

Steps 1–4 require time $O(m)$. Step 5 creates V_2' by distinguishing certain members of a previously defined set and requires time $O(m)$. Step 6 is simply an ordering of the $m+1$ activities according to the algorithm for forming basic cycles. Therefore, the time complexity of algorithm CCell is $O(m)$.

EXAMPLE 3.8 This example illustrates Steps 5 and 6. Suppose $m = 10$, $\delta = 5$, $\epsilon = 1$, and the vector of processing times is $p = (2, 4, 10, 1, 3, 2, 7, 4, 2, 3)$. Then $D_\delta = \{3, 7\}$, so $V_2' = \{3, 7\} \cup \{4, 8\} \cup \{2, 6\} = \{2, 3, 4, 6, 7, 8\}$.

$$\begin{aligned} \pi_B' &= (A_0, A_1, A_8, A_9, A_{10}, A_7, A_6, A_4, A_5, A_3, A_2), \\ T_c(\pi_B') &= \{18\delta + 22\epsilon + p_1 + p_5 + p_9 + p_{10}, \\ &\quad \max_{i\in V_2'}\{p_i + 3\delta + 4\epsilon + (r_i + q_i)(\delta + 2\epsilon) + \sum_{j\in X_i\cup Y_i} p_j\}\} \\ &= \max\{122, 32, 29, 30, 31, 26, 42\} = 122. \end{aligned}$$

THEOREM 3.20 *Algorithm CCell is a 1.5-approximation algorithm for the optimum per unit cycle time, and this bound is tight. Consequently, we have a 1.5-approximation for an optimal multi-unit cyclic solution.*

Proof.

a) From Theorem 3.19, if $p_i \leq \delta$, $\forall i$, then we have $\Omega_c \geq 2(m+1)\epsilon + \sum_{i=1}^{m} p_i + (m+2)\delta = T_c(\pi_U)$. Therefore, π_U is optimal.

b) If $\max_{1\leq i\leq m} p_i + 3\delta + 4\epsilon \geq 2(m+1)(\delta+\epsilon)$, then, by Lemma 3.6, π_D is optimal.

c) If $p_i \geq \delta$, $\forall i$, then, by Theorem 3.19, $\Omega_c \geq \max\{2(m+1)(\delta+\epsilon), \max_{1\leq i\leq m} p_i + 3\delta + 4\epsilon\} = T_c(\pi_D)$. Therefore, π_D is optimal.

d) If $|D_\delta| \geq \frac{1}{3}(m-2)$, then, from inequality (3.23), we have

$$\begin{aligned}\Omega_c &\geq 2(m+1)\epsilon + \sum_{i\in D_\delta^c} p_i + [m+2+\frac{1}{3}(m-2)]\delta \\ &\geq 2(m+1)\epsilon + \left(\frac{4}{3}m + \frac{4}{3}\right)\delta.\end{aligned}$$

Thus,

$$T_c(\pi_D) = 2(m+1)(\delta+\epsilon) \leq \frac{2(m+1)(\delta+\epsilon)}{2(m+1)\epsilon + \left(\frac{4}{3}m+\frac{4}{3}\right)\delta}\Omega_c \leq 1.5\Omega_c.$$

e) For $|D_\delta| < \frac{1}{3}(m-2)$, consider the 1-unit basic cycle π_B formed by the initial partition (V_1, V_2), where $V_1 = D_\delta^c$ and $V_2 = D_\delta$. Recall that the cycle time of π_B is $T_c(\pi_B) = \max\{\alpha, \beta_i | i \in V_2\}$, where $\alpha = 2(m+1)(\delta+\epsilon) + \sum_{i\in D_\delta^c}(p_i - \delta)$ and $\beta_i = p_i + 3\delta + 4\epsilon + (r_i + q_i)(\delta + 2\epsilon) + \sum_{j\in X_i\cup Y_i} p_j$.

An example of a 16-machine cell is shown in Figure 3.7, with machines corresponding to elements of D_δ and D_δ^c distinguished.

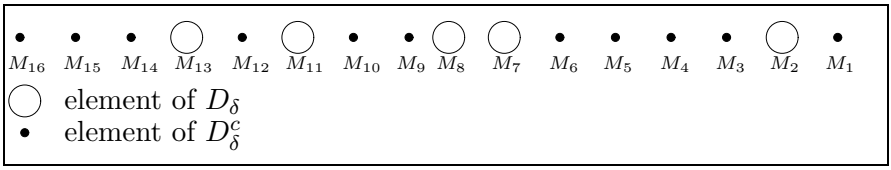


Figure 3.7. Machines Distinguished by $p_i \geq \delta$ or $p_i < \delta$.

For the basic cycle π'_B,

$$T_c(\pi'_B) = \max\{\alpha', \beta'_i | i \in V'_2\}, \tag{3.26}$$

where $\alpha' = 2(m+1)(\delta+\epsilon) + \sum_{i\in V_1'}(p_i - \delta)$, β_i' is defined as was β_i, and

$$\begin{aligned} V_2' &= D_\delta \cup \{j : j \in D_\delta^c,\ j = i+1,\ i \in D_\delta\} \cup \\ &\quad \{j : j \in D_\delta^c,\ j = i-1,\ i \in D_\delta\}. \end{aligned} \tag{3.27}$$

The same 16-machine cell is shown in Figure 3.8, with elements of V_1' and V_2' distinguished.

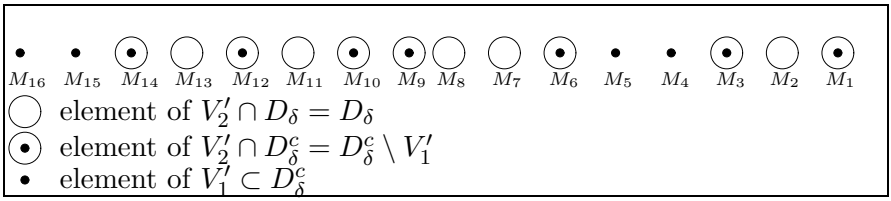


Figure 3.8. Machines Distinguished by Full Waiting or Partial Waiting in π_B'.

To estimate α', we first derive an inequality for later use. It follows immediately from the definition of V_2' in equation (3.27) that $|V_2'| \leq 3|D_\delta|$. Hence, $m - |V_2'| \geq m - 3|D_\delta|$. Since $|V_1^{'}| = m - |V_2^{'}|$ and $|D_\delta^c| = m - |D_\delta|$, we have $|V_1'| \geq |D_\delta^c| - 2|D_\delta|$, which implies

$$2|D_\delta| \ \geq\ |D_\delta^c| - |V_1'| = |D_\delta^c \setminus V_1'|, \tag{3.28}$$

in view of $V_1' \subset D_\delta^c$. The equality

$\alpha' = 2(m+1)(\delta+\epsilon) + \sum_{i\in V_1'}(p_i - \delta)$ implies that

$$\begin{aligned} \alpha' - \alpha &= \sum_{i\in V_1'}(p_i - \delta) - \sum_{i\in D_\delta^c}(p_i - \delta), \text{ so} \\ \alpha' &= \alpha + \sum_{i\in D_\delta^c}(\delta - p_i) - \sum_{i\in V_1'}(\delta - p_i) \\ &= \alpha + \sum_{i\in D_\delta^c \setminus V_1'}(\delta - p_i). \end{aligned}$$

From inequality (3.28), we have

$$\alpha' \ \leq\ \alpha + 2|D_\delta|\delta = \left(1 + \frac{2|D_\delta|\delta}{\alpha}\right)\alpha,$$

which, together with inequality (3.25), implies

$$\begin{aligned}
\alpha' &\leq \left(1 + \frac{2|D_\delta|\delta}{2(m+1)(\delta+\epsilon) + \sum_{i\in D_\delta^c}(p_i - \delta)}\right)\Omega_c \\
&\leq \left(1 + \frac{2|D_\delta|\delta}{2(m+1)(\delta+\epsilon) - |D_\delta^c|\delta)}\right)\Omega_c \\
&\leq \left(1 + \frac{2|D_\delta|}{2(m+1) - m + |D_\delta|}\right)\Omega_c \\
&= \left(1 + \frac{2|D_\delta|}{m+2+|D_\delta|}\right)\Omega_c.
\end{aligned}$$

This expression is increasing in $|D_\delta|$. Hence, $|D_\delta| < \frac{1}{3}(m-2)$ implies

$$\begin{aligned}
\alpha' &< \left(1 + \frac{\frac{2}{3}(m-2)}{m+2+\frac{1}{3}(m-2)}\right)\Omega_c = \left(1 + \frac{2m-4}{4m+4}\right)\Omega_c \\
&\leq 1.5\Omega_c. \qquad (3.29)
\end{aligned}$$

We now investigate the value of $\beta_i', i \in V_2'$:

1. By construction, for $i \in D_\delta$, $\beta_i' = p_i + 3\delta + 4\epsilon$. By Theorem 3.19, if $T_c(\pi_B') = p_i + 3\delta + 4\epsilon$ for some i, then π_B' is optimal.
2. For $i \in V_2' \setminus D_\delta$, $\beta_i' = p_i + 3\delta + 4\epsilon + (r_i + q_i)(\delta + 2\epsilon) + \sum_{j\in X_i\cup Y_i} p_j$. Since $i \in D_\delta^c$, $X_i \subset D_\delta^c$, and $Y_i \subset D_\delta^c$, we know that

$$p_i + \sum_{j\in X_i\cup Y_i} p_j \leq \sum_{j\in D_\delta^c} p_j.$$

π_B' was designed so that $|V_2'| \geq 2$. This implies that $r_i + q_i \leq m-2$. Hence,

$$\begin{aligned}
\beta_i' &\leq \sum_{j\in D_\delta^c} p_j + 3\delta + 4\epsilon + (m-2)(\delta + 2\epsilon) \\
&= \sum_{j\in D_\delta^c} p_j + (m+1)\delta + 2m\epsilon,\ i \in V_2'\setminus D_\delta.
\end{aligned}$$

This value is strictly less than α (defined in equation (3.24)), which, by inequality (3.25), is a lower bound on the cycle time. Hence, $\beta_i', i \in V_2' \setminus D_\delta$, will not dominate the cycle time expression in equation (3.26).

To summarize, for $i \in D_\delta$, $\beta'_i = p_i + 3\delta + 4\epsilon$ and for $i \in V'_2 \setminus D_\delta$, $\beta'_i < \alpha'$. Therefore, either $T_c(\pi'_B) = \alpha'$, or π'_B is optimal. Consequently, from inequality (3.29), π'_B is a 1.5 approximation for Ω_c. This completes the proof of the proposed bound.

We now demonstrate the tightness of the bound for $|D_\delta| \geq (m-2)/3$. Suppose $m = 8, \delta = 2, \epsilon = 0, p = (0, 0, 4, 0, 0, 4, 0, 0)$. Then

$$\begin{aligned} T_c(\pi_B) &= \max\{2(m+1)\delta - 6\delta, p_3 + 7\delta, p_6 + 7\delta\} \\ &= \max\{24, 18, 18\} = 24 = \Omega_c. \\ T_c(\pi_D) &= \max\{36, 10\} = 36. \end{aligned}$$

Hence, $T_c(\pi_D)/\Omega_c = 1.5$.

For $|D_\delta| < (m-2)/3$, we show the asymptotic tightness of the bound by taking a large m, the maximum possible $|D_\delta|$ for this m, and the maximum $|V'_2|$ $(= 3|D_\delta|)$. Let $m = 3k+3, k \in \mathbb{Z}^+; \delta = 1, \epsilon = 0; p_i = 4$ if $i = 3j, j = 1, \ldots, k$; $p_i = 0$ otherwise. Also, $D_\delta = \{3j : j = 1, \ldots, k\}, |V'_2| = 3k$, and $|D^c_\delta \setminus V'_1| = 2k$. Thus, by inequality (3.25),

$$\begin{aligned} \Omega_c \geq \alpha &= 2(3k+4)\delta - (2k+3)\delta = (4k+5)\delta \\ \alpha' &= \alpha + |D^c_\delta \setminus V'_1|\delta \\ &= (4k+5)\delta + 2k\delta = (6k+5)\delta, \text{ so} \\ \frac{\alpha'}{\Omega_c} &\leq \frac{6k+5}{4k+5} \to 1.5 \text{ as } k \to \infty. \end{aligned}$$

■

COROLLARY 3.6 *In a constant travel-time cell, a k-unit cyclic solution obtained by repeating an optimal 1-unit cyclic solution k times is a 1.5-approximation to an optimal k-unit cyclic solution, $k \geq 1$.*

3.5.3 Euclidean Travel-Time Cells

In this section we consider Euclidean cells. We start with elementary properties, and then provide a 4-approximation algorithm. We then provide an algorithm whose performance guarantee depends on the range of the inter-machine robot movement times. In some cases, this guarantee is better than 4. Once more, we begin by establishing a lower bound for the per unit cycle time.

THEOREM 3.21 *For any k-unit cycle π, $k \geq 1$, in a Euclidean travel-time cell, the per unit cycle time $T_e(\pi)/k$ satisfies*

$$\frac{T_e(\pi)}{k} \geq \max\left\{2(m+1)\epsilon + \sum_{i=0}^{m}\delta_{i,i+1} + \sum_{i=1}^{m}\min\{p_i, \min_{j\neq i}\{\delta_{j,i}\}\} + \min_{j\geq 1}\{\delta_{j,0}\}, \max_{1\leq i\leq m}\{p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon\}\right\}.$$

Proof. As before when proving Theorem 3.19, we establish a lower bound for the cycle time, and the result for the per unit cycle time follows immediately.

Consider the first argument. Each of the $k(m+1)$ activities requires one loading and one unloading, so the total time for these actions is $2k(m+1)\epsilon$. The second term represents the robot's total time for moves while performing the $k(m+1)$ activities. Before each time a machine $M_i, i = 1, \ldots, m$, is unloaded, there will be time taken either by a processing time (p_i) or a robot move (which takes at least time $\min_{j\neq i}\{\delta_{j,i}\}$). The robot never has to wait for processing to complete before getting a new part at I (unloading M_0), so the minimum time taken after all k such loadings is $k\min_{j\geq 1}\{\delta_{j,0}\}$.

For the second argument, observe that the sequence of actions between the start of M_i's unloading and the completion of its next loading must include activity A_i ($\delta_{i,i+1} + 2\epsilon$), travel by some route from M_{i+1} to M_{i-1} (the triangle inequality implies that the minimum possible time is $\delta_{i+1,i-1}$), and activity A_{i-1} ($\delta_{i-1,i} + 2\epsilon$). At minimum, this time is $\delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon$. Thus, the minimum time between each loading of M_i is $p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon$. In a k-unit cycle, this must be done k times. ∎

For convenience and consistency, we denote the per unit cycle time of an optimal k-unit cycle ($k \geq 1$) in a Euclidean travel-time cell by Ω_e.

LEMMA 3.7 *In a Euclidean robotic cell, the cycle time of the 1-unit reverse cycle is*

$$T_e(\pi_D) = \max\left\{2(m+1)\epsilon + \sum_{i=0}^{m}\delta_{i,i+1} + \sum_{i=1}^{m}\delta_{i+1,i-1} + \delta_{1,m}, \max_{1\leq i\leq m}\{p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon\}\right\}.$$

Proof. In the first argument, the total of loading and unloading times and the movement time during all activities is represented by the first two terms. Activity A_i completes when machine M_{i+1} is loaded. The robot's next action is to prepare to begin activity A_{i-1} by moving to $M_{i-1}, i = 1, \ldots, m$. The sum of these movement times is $\sum_{i=1}^{m} \delta_{i+1,i-1}$. After completing A_0, the robot moves to M_m to begin A_m ($\delta_{1,m}$).

In the second argument, if the robot waits at any machine M_i, the time spent between the completion of M_i's loading and the start of its unloading, obviously, is p_i. The duration between the completion of processing of one part and the beginning of processing of the next part is represented by the activity subsequence A_iA_{i-1}, which requires $\delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon$. ∎

COROLLARY 3.7 *If* $\max_{1\le i\le m}\{p_i+\delta_{i,i+1}+\delta_{i+1,i-1}+\delta_{i-1,i}+4\epsilon\} \ge 2(m+1)\epsilon + \sum_{i=1}^{m} \delta_{i,i+1} + \sum_{i=1}^{m} \delta_{i+1,i-1} + \delta_{1,m}$, *then* π_D *is optimal.*

Proof. By Lemma 3.7, the premise implies that $T_e(\pi_D) = \max_{1\le i\le m}\{p_i + \delta_{i,i+1}+\delta_{i+1,i-1}+\delta_{i-1,i}+4\epsilon\}$, which, by Theorem 3.21, is a lower bound for Ω_e. Thus, π_D is optimal. ∎

LEMMA 3.8 *In a Euclidean travel-time cell, a k-unit cyclic solution obtained by repeating the 1-unit reverse cycle k times provides a 4-approximation to the optimal k-unit cycle time,* $k \ge 1$.

Proof. If $T_e(\pi_D) = \max_{1\le i\le m}\{p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon\}$, then π_D is optimal. Otherwise, by using the triangle inequality, we get

$$\begin{aligned} T_e(\pi_D) &= 2(m+1)\epsilon + \sum_{i=0}^{m} \delta_{i,i+1} + \sum_{i=1}^{m} \delta_{i+1,i-1} + \delta_{1,m} \\ &\le 2(m+1)\epsilon + \sum_{i=0}^{m} \delta_{i,i+1} + \left(\sum_{i=0}^{m-1} \delta_{i,i+1} + \sum_{i=1}^{m} \delta_{i,i+1} \right) \\ &\quad + \sum_{i=1}^{m-1} \delta_{i,i+1} \\ &\le 2(m+1)\epsilon + 4\sum_{i=0}^{m} \delta_{i,i+1} \;\le\; 4\Omega_e. \end{aligned}$$

∎

For our next result, we use three constants: $\delta = \min_{0 \le i < j \le m+1}\{\delta_{i,j}\}$, $\Delta = \max_{0 \le i < j \le m+1}\{\delta_{i,j}\}$, and $q = \Delta/\delta$.

Algorithm ECell provides a $1.5q$ bound for the optimum per unit cycle time. Since finding an optimal 1-unit cycle for Euclidean cells is NP-hard (Brauner et al. [24]), such an efficient approximation can be very useful. If $q < 4/1.5 \approx 2.67$, then the algorithm yields a tighter bound than 4. Intuitively, if the variation in the inter-machine travel times is not large, a reasonable approximation is to assume that the Euclidean cell is a constant travel-time cell. This is the main idea behind algorithm ECell.

Algorithm ECell

Input: The data for a Euclidean travel-time simple robotic cell: m, $\delta_{i,j}, 0 \le i < j \le m+1$, ϵ, $p_i, i = 1, \ldots, m$.

Step 1: If $p_i \le \delta$, $\forall i$, then output $\pi_U = (A_0, A_1, \ldots, A_m)$. Stop.

Step 2: If $\max_{1 \le i \le m}\{p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon\} \ge 2(m+1)\epsilon + \sum_{i=1}^{m} \delta_{i,i+1} + \sum_{i=1}^{m} \delta_{i+1,i-1} + \delta_{1,m}$, then output $\pi_D = (A_0, A_m, A_{m-1}, \ldots, A_2, A_1)$. Stop.

Step 3: If $p_i \ge \delta$, $\forall i$, then output π_D. Stop.

Step 4: If $q \ge 4/1.5 \approx 2.67$, then output π_D. Stop.

Step 5: Define the set $D_\Delta = \{i : p_i \ge \Delta\}$. If $|D_\Delta| \ge (m-2)/3$, then output π_D. Stop.

Step 6: If $|D_\Delta| < (m-2)/3$, then form the set V_2' as follows:

$$\begin{aligned} V_2' &= D_\Delta \cup \{j : j \in D_\Delta^c,\ j = i+1,\ i \in D_\Delta\} \\ &\quad \cup \{j : j \in D_\Delta^c,\ j = i-1,\ i \in D_\Delta\}. \end{aligned}$$

Step 7: Form basic cycle π_B' such that machines corresponding to the elements of V_2' have partial waiting and machines corresponding to the elements of $V_1' = M \backslash V_2'$ have full waiting. Output π_B'. Stop.

Steps 1–5 require time $O(m)$. Step 6 creates V_2' by distinguishing certain members of a previously defined set and requires time $O(m)$. Step 7 is simply an ordering of the $m+1$ activities according to the

algorithm for forming basic cycles. The time complexity of algorithm ECell is thus $O(m)$.

THEOREM 3.22 *Algorithm ECell is a* $1.5q$*-approximation algorithm for the optimum per unit cycle time, and this bound is tight. Consequently, we have a* $1.5q$*-approximation for an optimal multi-unit cyclic solution.*

Proof. Consider two related constant travel-time cells whose machine processing times $p_i, i = 1, \ldots, m$, are the same as those of the Euclidean cell. One of these new cells will have constant travel time δ, and the other will have constant travel time Δ. Their respective optimum per unit cycle times are Ω_δ and Ω_Δ. Clearly, $\Omega_\delta \leq \Omega_e \leq \Omega_\Delta$.

a) From Theorem 3.19, if $p_i \leq \delta$, $\forall i$, then we have $\Omega_e \geq \Omega_\delta \geq 2(m+1)\epsilon + (m+2)\delta + \sum_{i=1}^{m} p_i$. In the Euclidean cell,

$$T_e(\pi_U) \;=\; 2(m+1)\epsilon + \sum_{i=0}^{m} \delta_{i,i+1} + \delta_{m+1,0} + \sum_{i=1}^{m} p_i.$$

Since $\Delta = \max_{0 \leq i < j \leq m+1}\{\delta_{i,j}\}$, we have

$$\begin{aligned} T_e(\pi_U) &\leq 2(m+1)\epsilon + (m+2)\Delta + \sum_{i=1}^{m} p_i \\ &\leq \frac{2(m+1)\epsilon + (m+2)\Delta + \sum_{i=1}^{m} p_i}{2(m+1)\epsilon + (m+2)\delta + \sum_{i=1}^{m} p_i}\Omega_e \\ &\leq \frac{\Delta}{\delta}\Omega_e = q\Omega_e. \end{aligned}$$

b) If $\max_{1 \leq i \leq m}\{p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon\} \geq 2(m+1)\epsilon + \sum_{i=1}^{m} \delta_{i,i+1} + \sum_{i=1}^{m} \delta_{i+1,i-1} + \delta_{1,m}$, then by Corollary 3.7, π_D is optimal.

c) If $\max_{1 \leq i \leq m}\{p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon\} < 2(m+1)\epsilon + \sum_{i=1}^{m} \delta_{i,i+1} + \sum_{i=1}^{m} \delta_{i+1,i-1} + \delta_{1,m}$, then by Lemma 3.7, $T_e(\pi_D) \leq 2(m+1)(\Delta + \epsilon)$.

If $p_i \geq \delta$, $\forall i$, then by Theorem 3.21, $\Omega_e \geq 2(m+1)(\delta+\epsilon)$. Therefore,

$$\begin{aligned} T_e(\pi_D) &\leq \frac{2(m+1)(\Delta+\epsilon)}{2(m+1)(\delta+\epsilon)}\Omega_e \\ &\leq \frac{\Delta}{\delta}\Omega_e = q\Omega_e. \end{aligned}$$

d) If $q \geq 4/1.5$, then $4 \leq 1.5q$, and so $T_e(\pi_D) \leq 1.5q\Omega_e$.

e) By Theorem 3.20, algorithm CCell applied to the constant travel-time Δ cell can provide a 1.5-approximation for Ω_Δ. Denoting the value of this heuristic by $H_c(\Omega_\Delta)$, we have by Theorem 3.19,

$$\begin{aligned} H_c(\Omega_\Delta) &\leq 1.5[(m+2)\Delta + \sum_{i=1}^{m} \min\{p_i, \Delta\} + 2(m+1)\epsilon] \\ &= 1.5[(m+2)q\delta + \sum_{i=1}^{m} \min\{p_i, q\delta\} + 2(m+1)\epsilon] \\ &\leq 1.5q[(m+2)\delta + \sum_{i=1}^{m} \min\{p_i, \delta\} + 2(m+1)\epsilon] \\ &\leq 1.5q\Omega_\delta. \end{aligned}$$

Since $H_c(\Omega_\Delta) \leq 1.5q\Omega_\delta \leq 1.5q\Omega_e$, algorithm CCell executed on the constant travel-time cell with travel time $\Delta = \max_{0\leq i<j\leq m+1}\{\delta_{i,j}\}$ provides a $1.5q$ bound for the optimum per unit cycle time in a Euclidean cell.

To prove asymptotic tightness for the case in which $|D_\delta| \geq (m-2)/3$, we first need an expression for the cycle time of a basic cycle π_B in a Euclidean cell: $T_e(\pi_B) = \max\{\alpha, \beta_i | i \in V_2\}$, where

$$\begin{aligned} \alpha &= 2(m+1)\epsilon + \sum_{i=0}^{m} \delta_{i,i+1} + \sum_{i\in V_1} p_i + \sum_{i=0}^{n-1} \delta_{v_{i+2},v_i} + \delta_{v_1,v_n}, \\ V_2 &= \{v_1, v_2, \ldots, v_n\}, v_0 = 0, v_{n+1} = m+1, \text{ and} \\ \beta_i &= p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon + \sum_{j\in X_i\cup Y_i} (p_j + \delta_{j,j+1} + 2\epsilon). \end{aligned}$$

Let $m = 3k+2, k \in \mathbb{Z}^+; \epsilon = 0; p_{3j} = 2\Delta, j = 1, \ldots, k$, and $p = 0$ otherwise. Let $\delta_{i+1,i-1} = \Delta, i = 1, \ldots, m, \delta_{1,m} = \Delta$, $\delta_{i,j} = \delta$ otherwise; and $q \leq 2$. If the basic cycle formed from the initial partition is π_B, then $\alpha = (3k+3)\delta + (k+1)\delta = 4(k+1)\delta$. Thus,

$$\begin{aligned} T_e(\pi_B) &= \max\{4(k+1)\delta, 3\Delta + 6\delta\} = 4(k+1)\delta = \Omega_e, \\ T_e(\pi_D) &= (3k+3)(\delta + \Delta). \end{aligned}$$

Hence,

$$\frac{T_e(\pi_D)}{\Omega_e} = \frac{(3k+3)(\delta+\Delta)}{4(k+1)\delta} = \frac{3}{4}(1+q) \to 1.5 \text{ as } q \to 1.$$

For $|D_\delta| < (m-2)/3$, let $m = 3k+3, k \in \mathbb{Z}^+; \epsilon = 0; p_{3j} = 2\Delta, j = 1,\ldots,k, p = 0$ otherwise. Then, $D_\delta = \{3j : j = 1,2,\ldots,k\}$. Let $\delta_{i,i+1} = \delta, i = 0,\ldots,3k+3$, $\delta_{3j,3j-6} = \delta, j = 2,3,\ldots,k$, $\delta_{3,3k} = \delta$, and $\delta_{i,j} = \Delta$, otherwise. Therefore,

$$\begin{aligned} \Omega_e = \alpha &= (3k+4+k+1)\delta = (4k+5)\delta, \\ \alpha' &= (3k+3)\delta + (3k+2)\Delta, \\ \frac{\alpha'}{\Omega_e} &= \frac{(3k+3)+(3k+2)q}{4k+5} \rightarrow 1.5 \text{ as } q \rightarrow 1 \text{ and } k \rightarrow \infty. \end{aligned}$$

■

COROLLARY 3.8 *In a Euclidean travel-time cell, a k-unit cyclic solution obtained by repeating an optimal 1-unit cyclic solution k times provides a $1.5q$-approximation to an optimal k-unit cyclic solution, $k \geq 1$.*

To summarize, we have $O(m)$ algorithms that produce cyclic solutions whose per unit cycle times are within a constant factor of the optimum for the three most common classes of robotic cells viz., additive, constant, and Euclidean travel-time. The approximation guarantees for these three classes of cells are 10/7, 1.5, and 4, respectively. Note that the approximation algorithms construct a cyclic solution by repeating a 1-unit cycle. Therefore, the guarantees are, in fact, upper bounds on ratio of the per unit cycle time of an optimal 1-unit cycle to that of an optimal multi-unit cycle for each of these three classes of cells.

Chapter 4

DUAL-GRIPPER ROBOTS

This chapter treats the problem of sequencing robot moves in a dual-gripper robotic cell producing a single part-type. In cells with a single-gripper robot, a part cannot be moved from its current machine to the next one if the next one is occupied (this is referred to as *blocking* in classical flow shop scheduling; see Chapter 2). However, this is possible with a dual-gripper robot because it can hold two parts simultaneously. The grippers reside at the end of the robot's arm. In a typical usage, one gripper is empty and the other holds a part to be loaded onto the next intended machine, which is currently occupied. The robot moves its arm to that machine, uses the empty gripper to unload the finished part, rotates the "wrist" at the end of its arm, and loads the other part. The dual-gripper robot can perform such an unloading-and-loading sequence faster than a single-gripper robot because the rotation takes less time than does the robot's inter-machine movement. Hence, dual-gripper robots can increase throughput in cells that are constrained by the robot's speed. The combinatorial explosion in the number of possible robot moves with a dual-gripper cell substantially complicates the analytical and computational aspects of the sequencing problem as compared to those in a single-gripper cell.

Our presentation in this chapter reflects the different types of cells that have been considered in the various studies on dual-gripper robotic cells. Some model the additive travel metric, while others the constant travel metric; some have addressed cells with a circular layout with the

input device I and the output device O at the same location (Figure 4.1), whereas others use a linear, semicircular, or circular layout with separate locations for I and O. In switching between these models, our intention is to remain true to the original research.

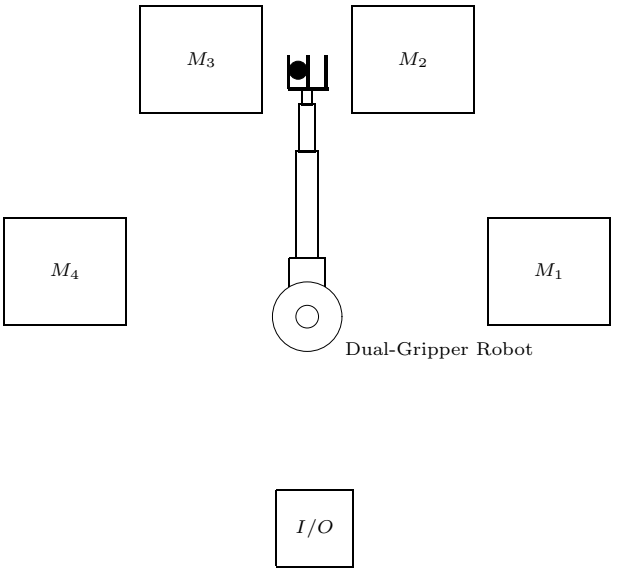


Figure 4.1. A Four-Machine Dual-Gripper Robot Cell with the Input and Output Devices at the Same Location.

4.1 Additional Notation

We let θ denote the gripper switch time – the duration between the moment one gripper has unloaded a machine until the moment the second gripper is positioned to load the same machine. Furthermore, we assume that a robot gripper switch is required (if I and O are co-located) when the robot executes the operation "pick up a part from the input device" followed by the operation "drop a part onto the output device"; the same two operations performed in the reverse order (i.e., "drop a part onto the output device" followed by "pick up a part from the input device") do not require a gripper switch. This is because in the latter case the same gripper that drops a part onto the output device can be then used to pick up a part from the input device.

Moreover, we assume that the time required for the dual-gripper robot to reposition its grippers while traveling from one machine to another (i.e., when two successive operations are executed on different machines and require different grippers) does not exceed the robot travel time between the machines. Formally, let $\bar{\theta}$ denote the time taken by the robot to switch its grippers while traveling between machines. For all the problems considered in this chapter, we assume that $\bar{\theta} \leq \delta$ so that $\max\{\delta, \bar{\theta}\} = \delta$; this assumption is valid in most manufacturing applications. We will use the following additional notation.

- $P_j \rightarrow R$: Part P_j is transferred to the robot from a machine (i.e., the machine is unloaded).
- $R \rightarrow M_i$: The robot moves from its last location to M_i and positions one of its grippers for the next load or unload operation.
- $P_j \rightarrow M_i$: Part P_j is transferred from the robot to M_i (i.e., M_i is loaded)
- *wait*: The robot waits at some machine M_i for completing the processing of a part P_j.

Venkatesh et al. [154] study ways to improve the throughput of an additive travel-time cluster tool in semiconductor fabrication that uses a dual-gripper robot. They show that a dual-gripper robot improves throughput over a single-gripper robot when part processing times dominate robot travel times. They also show that the travel speeds of a single-gripper robot would have to be twice those of a dual-gripper robot to achieve the same throughput. One of our objectives is to compare the maximum throughputs of single-gripper and dual-gripper cells in a more general framework. Throughout this chapter, we consider cells with a circular layout. Sections 4.2–4.7 consider the problem of obtaining an optimum 1-unit cycle in an m-machine dual-gripper cell under the additive travel-time metric (i.e., $RF_m^{2,\circ}|(\textit{free,A,cyclic-1})|\mu$); Sections 4.3–4.6 present the results from Sethi et al. [143]. Section 4.8 studies a robotic cell model that is closely related to a dual-gripper cell. Specifically, we consider throughput maximization for a robotic cell served by a single-gripper robot with a unit-capacity output buffer at each machine (i.e., $RF_{m,\bar{1}}^{1,\circ}|(\textit{free,A,cyclic-1})|\mu$). Cells with the constant travel-time metric

are discussed in Section 4.9. The input device I and the output device O are assumed to be co-located in Sections 4.2–4.8, and are considered to be in separate locations in Section 4.9. To keep the description simple, we avoid the explicit specification of the circular layout in most of the notation used here. We start our analysis with two-machine cells.

4.2 Cells with Two Machines

The dual-gripper robot move sequencing problem is much more complicated than that with a single gripper even in the two-machine case. Su and Chen [149] were the first to consider a dual-gripper robotic cell with two machines producing a single part type ($RF_m^2|(free,A,cyclic\text{-}1)|\mu$). They consider only five different 1-unit cycles. We will see shortly that a complete search for optimality involves analyzing 52 cycles (Sethi et al. [143]). As a result, a new framework needs to be developed to characterize all possible cycles and their cycle time expressions in terms of the cell data, such as the processing times of parts on the machines, the robot travel time between two adjacent locations, and the robot load and unload times.

As with single-gripper cells, a complete specification of the state of a robotic cell would be unduly burdensome for specifying the robot move cycles of interest. The following robot states, which we refer to as *basic robot states*, are sufficient to specify the robot move cycles.

- $R_i^-(0,k)$ or $R_i^-(k,0)$ is the robot state in which the robot has just finished loading (or dropping off) a part on M_i, $i = 1,\ldots,m+1$, from a gripper. This gripper now holds zero parts as indicated by the argument which equals 0. The other gripper has no part if $k = 0$, and if $k \neq 0$, it has one part to be processed (or dropped off) on M_k, $k = 1,\ldots,m+1$.

- $R_i^+(i+1,k)$ or $R_i^+(k,i+1)$ is the state in which the robot has just finished unloading (or picking up) a part from M_i, $i = 0,\ldots,m$. This part is now held in a gripper to be processed (or dropped off) on M_{i+1}, where machine M_{m+1} means O. The meaning of the second argument k, $k = 0,1,\ldots,m+1$, is the same as before.

It should be clear that in each 1-unit cycle for each i ($1 \leq i \leq m+1$), there will be precisely one loading activity corresponding to a state

$R_i^-(\cdot,\cdot)$, where the dots will be replaced by appropriate indices; and one unloading activity corresponding to a state $R_i^+(\cdot,\cdot)$, where again the dots are replaced by appropriate indices. Thus, there are $2m+2$ such states in a 1-unit cycle (Sethi et al. [142]). Given these, a set of equations defining robot waiting times at different machines can be easily written. It is straightforward to derive the expressions for the cycle times from solutions of these equations.

Recall, Chapter 3, the following feasibility criteria for robot moves in a single-gripper cell:

- The robot cannot be instructed to load an occupied machine.
- The robot cannot be instructed to unload an unoccupied machine.

In addition to the above, we have the following necessary criteria for a cycle to be feasible in a dual-gripper cell:

- The robot cannot be instructed to unload a machine if both of its grippers are occupied, i.e., a feasible activity sequence cannot have a subsequence (M_i^+, M_j^+, M_k^+).
- The robot cannot be instructed to load a machine if both of its grippers are empty, i.e., a feasible activity sequence cannot have a subsequence (M_i^-, M_j^-, M_k^-).
- The robot can load a part onto M_i only if that part's most recent processing was on M_{i-1}, $i = 1, \ldots, m+1$.

To obtain all feasible 1-unit cycles for $m = 2$, a convenient set of basic robot states from which to begin the cycles is $R_2^+(3,k), 0 \le k \le 3$. A feasible sequence of operations is one that satisfies all of the operating constraints of the system, e.g., the parts are scheduled as a flow shop, the robot can hold at most two parts at a time, each machine can hold at most one part at a time, etc. We consider only feasible state sequences. For example, a sequence in which the state $R_2^+(3,2)$ is immediately followed by $R_1^-(0,k)$ is not feasible and can be eliminated: in the first state, $R_2^+(3,2)$, the robot has just unloaded a part from M_2 and is holding two parts that require processing next on M_2 and M_3. The next state $R_1^-(0,k)$ is not feasible immediately after the state $R_2^+(3,2)$, since loading a part onto M_1 is not possible as the robot does not have a part in either of its grippers that requires processing on M_1.

Before we discuss the generation of all 1-unit cycles, it is instructive to consider a detailed example of a 1-unit robot move cycle for $m = 2$. Consider the cycle $(R_2^+(3,2),\ R_2^-(3,0), R_3^-(0,0),\ R_3^+(1,0),\ R_1^+(1,2),\ R_1^-(0,2), R_2^+(3,2))$. At the beginning of the $(j-1)$th iteration, part P_{j-1} has been unloaded from M_2 and placed on the robot, P_j (already on the robot) awaits loading on M_2, and P_{j+1} is being processed at M_1. Using our previously defined notation, we may completely describe the cycle as in Table 4.1. The cycle time for this cycle is $3\delta + 6\epsilon + 2\theta + w_1 + w_2$ where, for now, we ignore the calculation of w_1 and w_2.

Cycle Element	Operations	Duration
$R_2^-(3,0)$	switch grippers	θ
	$P_j \rightarrow M_2$	ϵ
$R_3^-(0,0)$	$R \rightarrow I/O$	δ
	$P_{j-1} \rightarrow I/O$	ϵ
$R_3^+(1,0)$	$P_{j+2} \rightarrow R$	ϵ
$R_1^+(1,2)$	$R \rightarrow M_1$	δ
	wait	w_1
	$P_{j+1} \rightarrow R$	ϵ
$R_1^-(0,2)$	switch grippers	θ
	$P_{j+2} \rightarrow M_1$	ϵ
$R_2^+(3,2)$	$R \rightarrow M_2$	δ
	wait	w_2
	$P_j \rightarrow R$	ϵ

Table 4.1. A 1-Unit Cycle.

We now consider all possible (a total of 52) 1-unit cyclic sequences organized under four cases. Each of the four cases is specified by its initial basic robot state. For each case, we consider all possible cyclic sequences starting from the specified basic robot state. We will eventually show that of the 52 sequences listed, all but 13 are dominated. The sequences under Case $i, i = 1, 2, 3, 4$, have state $R_2^+(3, i-1)$ as the initial state. For an enumeration of the sequences under Cases 1, 2, and 3, we refer the reader to Figures 4.2, 4.3, and 4.4, respectively.

Case 4. (*sequences starting with the state* $R_2^+(3,3)$) At the start of the cycle, the robot holds two parts, both destined for M_3, the I/O location. Since a basic robot state of the form $R_3^-(\cdot,\cdot)$ can occur only once in a

cycle, it follows that the robot will carry one of these two parts throughout the whole cycle (in effect, the robot will be operating as if it were a single-gripper robot). Based on this observation, the reader can confirm that there are only two feasible cycles that start with $R_2^+(3,3)$, namely:

$$\begin{aligned} C_{4,1} &= (R_2^+(3,3), R_3^-(3,0), R_1^+(3,2), R_2^-(3,0), \\ &\quad R_3^+(3,1), R_1^-(3,0), R_2^+(3,3)), \end{aligned}$$

and

$$\begin{aligned} C_{4,2} &= (R_2^+(3,3), R_3^-(3,0), R_3^+(3,1), R_1^-(3,0), \\ &\quad R_1^+(3,2), R_2^-(3,0), R_2^+(3,3)). \end{aligned}$$

Notice that in terms of the sequence of loadings and unloadings, $C_{4,1}$ is equivalent to $C_{1,18}$, and $C_{4,2}$ is equivalent to $C_{1,13}$ (see Figure 4.2). Cycles $C_{4,1}$ and $C_{4,2}$, therefore, have the same cycle times as $C_{1,18}$ and $C_{1,13}$, respectively.

For problem $RF_2^2|(\textit{free,A,cyclic-1})|\mu$, the cycle time expressions for the 52 cycles can be easily derived and are shown in Tables 4.2, 4.3, 4.4, and 4.5. All but 13 of the cycles in Tables 4.2, 4.3, 4.4, and 4.5 are dominated. The last column in these tables indicates the dominance, if applicable. The 13 undominated cycles are: $C_{1,1}$, $C_{1,4}$, $C_{1,13}$, $C_{1,14}$, $C_{1,15}$, $C_{3,3}$, $C_{3,4}$, $C_{3,5}$, $C_{3,6}$, $C_{3,9}$, $C_{3,10}$, $C_{3,11}$, and $C_{3,16}$. It can be easily verified by examining the 13 undominated cycles that $C_{3,10}$ is optimal among all dual-gripper 1-unit cyclic schedules under the assumption that $\theta \leq \min\{\delta, p_1, p_2\}$.

Using the preceding analysis of two-machine cells as intuition, our next task is to generalize cycle $C_{3,10}$ for a general m-machine dual-gripper cell.

4.3 A Cyclic Sequence for m-Machine Dual-Gripper Cells

We now construct a generalization C_m^d of cycle $C_{3,10}$ for m-machine cells. We first analyze its cycle time and then show its optimality for problem $RF_m^2|(\textit{free,A,cyclic-1})|\mu$ under the assumption that $\theta \leq \min\{\delta, p_1, \ldots, p_m\}$.

Cycle C_d^m starts with all machines occupied by parts and the empty robot positioned at I/O. The sequence of activities for the robot in this cycle is as follows:

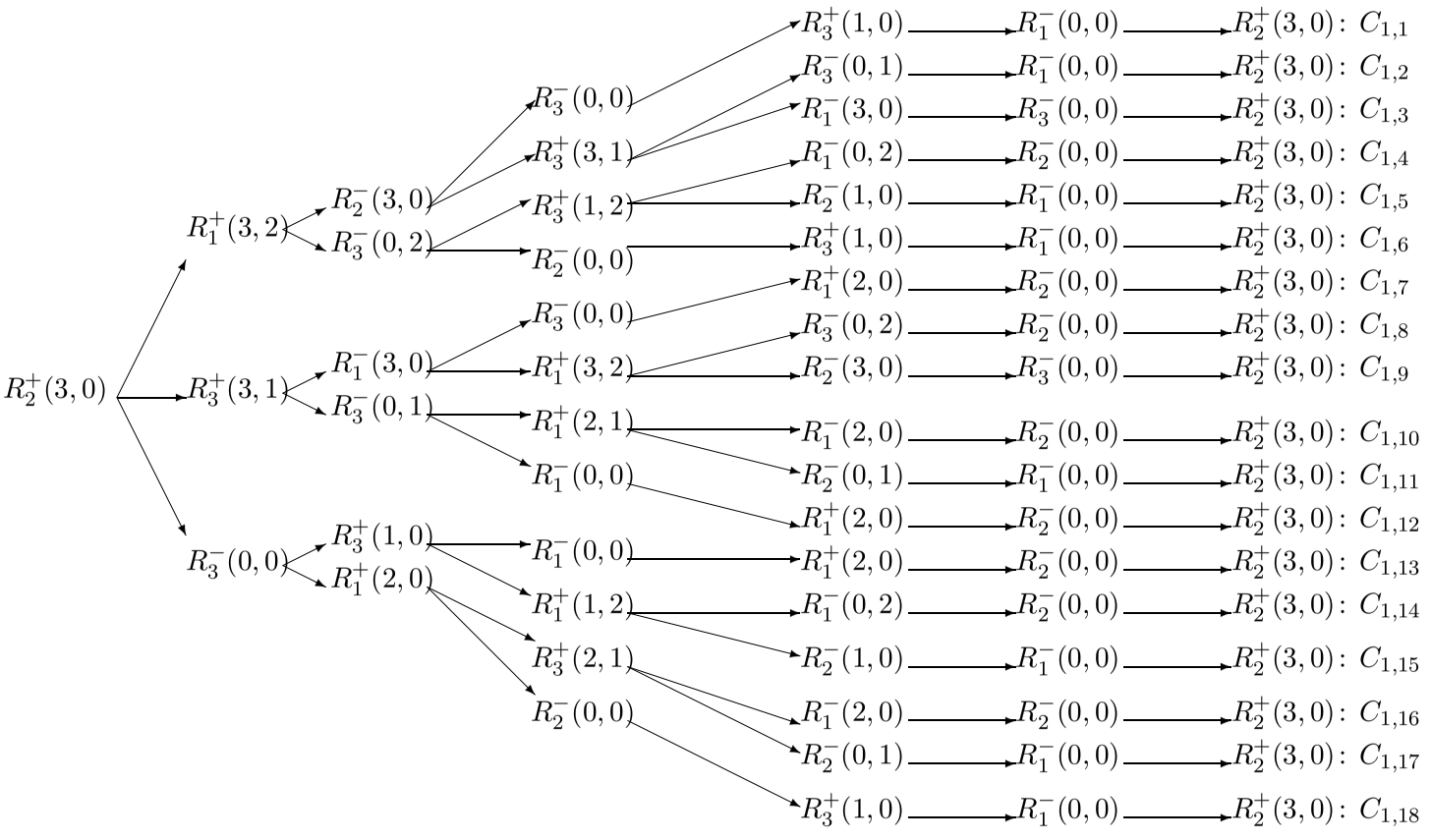


Figure 4.2. 18 Feasible Cycles for Case 1, Beginning with the State $R_2^+(3,0)$.

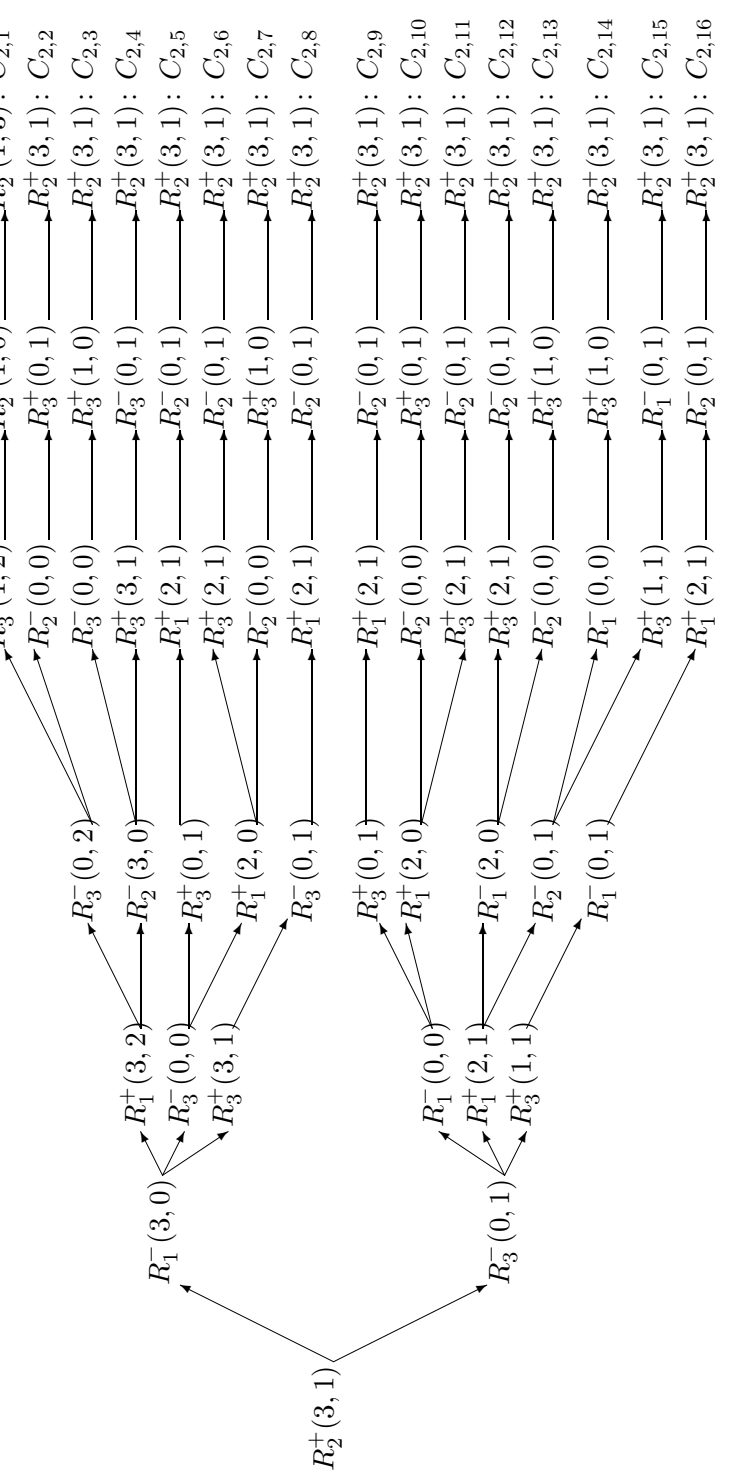


Figure 4.3. 16 Feasible Cycles for Case 2, Beginning with the State $R_2^+(3,1)$.

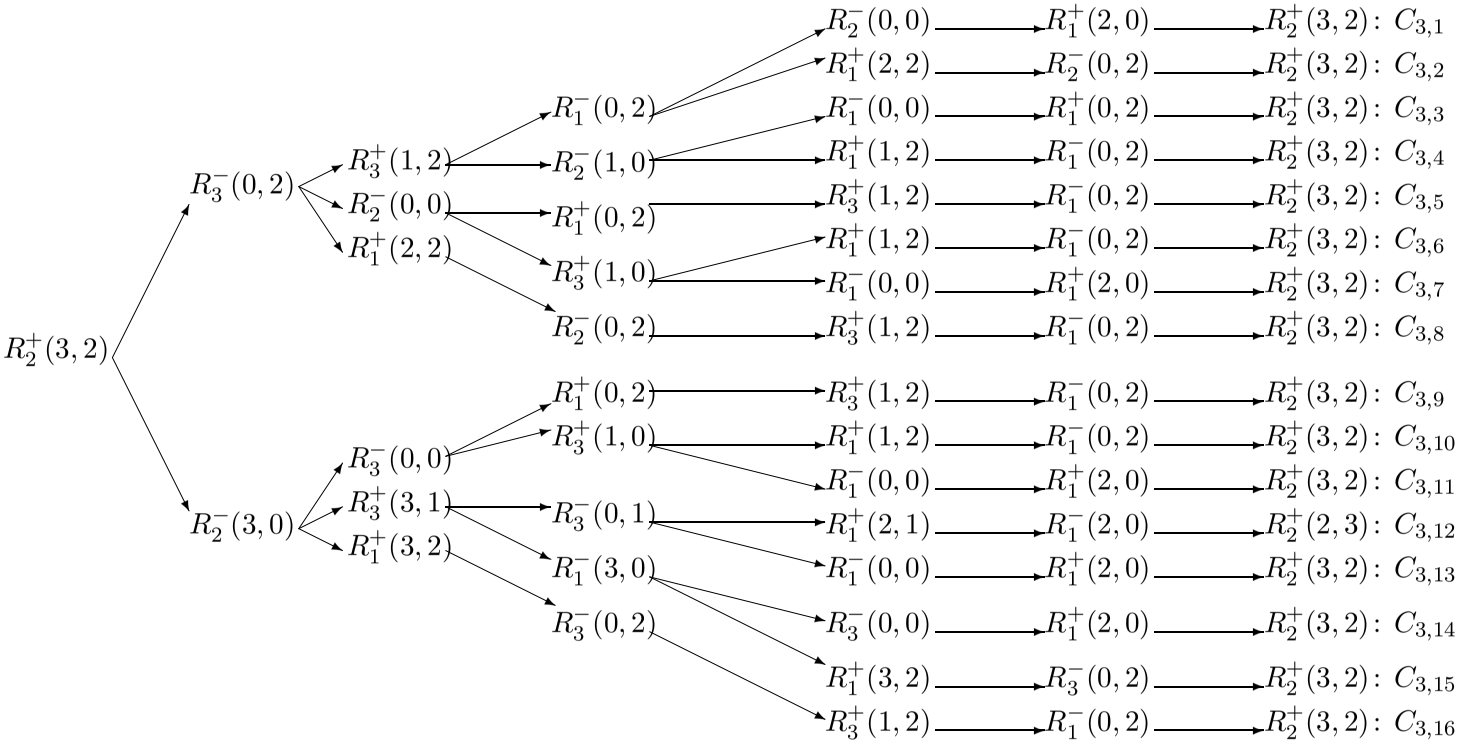


Figure 4.4. 16 Feasible Cycles for Case 3, Beginning with the State $R_2^+(3,2)$.

Cycle	Cycle Time	Dominance Relation
$C_{1,1}$	$T_{1,1} = \max\{5\delta + 6\epsilon, p_1 + 3\delta + 5\epsilon, p_2 + 2\delta + 3\epsilon\}$	undominated
$C_{1,2}$	$T_{1,2} = T_{1,1} + \theta$	$T_{1,2} \geq T_{1,1}$
$C_{1,3}$	$T_{1,3} = \max\{6\delta + 6\epsilon, p_1 + 3\delta + 4\epsilon, p_2 + 2\delta + 3\epsilon\}$	$T_{1,3} \geq T_{3,5}$
$C_{1,4}$	$T_{1,4} = \max\{p_1 + 2\delta + 4\epsilon, p_2 + 4\delta + 6\epsilon\}$	undominated
$C_{1,5}$	$T_{1,5} = \max\{5\delta + 6\epsilon, p_1 + 3\delta + 5\epsilon, p_2 + 3\delta + 5\epsilon\}$	$T_{1,5} \geq T_{1,1}$
$C_{1,6}$	$T_{1,6} = \max\{6\delta + 6\epsilon, p_1 + 4\delta + 5\epsilon, p_2 + 3\delta + 4\epsilon\}$	$T_{1,6} \geq T_{1,1}$
$C_{1,7}$	$T_{1,7} = \max\{p_2 + 5\delta + 6\epsilon, p_1 + p_2 + 3\delta + 5\epsilon\}$	$T_{1,7} \geq T_{1,14}$
$C_{1,8}$	$T_{1,8} = 4\delta + 6\epsilon + p_1 + p_2$	$T_{1,8} \geq T_{1,13}$
$C_{1,9}$	$T_{1,9} = \max\{p_1 + 5\delta + 6\epsilon, p_1 + p_2 + 3\delta + 5\epsilon\}$	$T_{1,9} \geq T_{1,1}$
$C_{1,10}$	$T_{1,10} = T_{1,14} + \theta$	$T_{1,10} \geq T_{1,14}$
$C_{1,11}$	$T_{1,11} = T_{1,15} + \theta$	$T_{1,11} \geq T_{1,15}$
$C_{1,12}$	$T_{1,12} = T_{1,13} + \theta$	$T_{1,12} \geq T_{1,13}$
$C_{1,13}$	$T_{1,13} = 3\delta + 6\epsilon + p_1 + p_2$	undominated
$C_{1,14}$	$T_{1,14} = \max\{p_1 + 2\epsilon + \theta, p_2 + 3\delta + 6\epsilon + \theta\}$	undominated
$C_{1,15}$	$T_{1,15} = \max\{5\delta + 6\epsilon, p_1 + 2\delta + 3\epsilon, p_2 + 3\delta + 5\epsilon\}$	undominated
$C_{1,16}$	$T_{1,16} = \max\{p_1 + 2\delta + 3\epsilon, p_2 + 5\delta + 6\epsilon\}$	$T_{1,16} \geq T_{1,15}$
$C_{1,17}$	$T_{1,17} = \max\{6\delta + 6\epsilon, p_1 + 3\delta + 4\epsilon, p_2 + 4\delta + 5\epsilon\}$	$T_{1,17} \geq T_{1,18}$
$C_{1,18}$	$T_{1,18} = \max\{6\delta + 6\epsilon, p_1 + 3\delta + 4\epsilon, p_2 + 3\delta + 4\epsilon\}$	$T_{1,18} \geq T_{1,3}$

Table 4.2. Cycle-Time Expressions for 1-Unit Cycles $C_{1,k}$ for $m = 2$.

Cycle C_d^m
Begin
  ϵ: Robot picks up a part from I/O.
  For $i = 1$ to m do:
  Begin
    δ: Robot moves to M_i.
    w_i: Robot waits for the part on M_i to be completed.
    ϵ: Robot unloads M_i.
    θ: Robot switches to the other gripper.
    ϵ: Robot loads M_i.
  End
  δ: Robot moves to I/O.
  ϵ: Robot unloads finished part at I/O.
End

Cycle	Cycle Time	Dominance Relation
$C_{2,1}$	$T_{2,1} = 3\delta + 6\epsilon + p_1 + p_2 = T_{1,13}$	$T_{2,1} \geq T_{1,13}$
$C_{2,2}$	$T_{2,2} = \max\{p_1 + 5\delta + 6\epsilon, p_1 + p_2 + 3\delta + 5\epsilon\}$	$T_{2,2} \geq T_{2,3}$
$C_{2,3}$	$T_{2,3} = \max\{p_1 + 4\delta + 6\epsilon, p_1 + p_2 + 2\delta + 4\epsilon\}$	$T_{2,3} \geq T_{3,3}$
$C_{2,4}$	$T_{2,4} = T_{2,3} + \theta$	$T_{2,4} \geq T_{2,3}$
$C_{2,5}$	$T_{2,5} = \max\{p_2 + 4\delta + 6\epsilon, p_1 + p_2 + 2\delta + 4\epsilon\}$	$T_{2,5} \geq T_{1,4}$
$C_{2,6}$	$T_{2,6} = \max\{p_2 + 5\delta + 6\epsilon, p_1 + p_2 + 3\delta + 5\epsilon\}$	$T_{2,6} \geq T_{2,5}$
$C_{2,7}$	$T_{2,7} = \max\{6\delta + 6\epsilon, p_1 + 4\delta + 5\epsilon, p_2 + 4\delta + 5\epsilon,$ $p_1 + p_2 + 2\delta + 4\epsilon\}$	$T_{2,7} \geq T_{2,14}$
$C_{2,8}$	$T_{2,8} = T_{2,5} + \theta$	$T_{2,8} \geq T_{2,5}$
$C_{2,9}$	$T_{2,9} = \max\{p_2 + 5\delta + 6\epsilon, p_1 + p_2 + 3\delta + 5\epsilon\}$	$T_{2,9} \geq T_{2,5}$
$C_{2,10}$	$T_{2,10} = \max\{p_1 + 5\delta + 6\epsilon, p_1 + p_2 + 3\delta + 5\epsilon\}$	$T_{2,10} \geq T_{2,3}$
$C_{2,11}$	$T_{2,11} = 4\delta + 6\epsilon + p_1 + p_2$	$T_{2,11} \geq T_{2,3}$
$C_{2,12}$	$T_{2,12} = \max\{p_1 + 2\epsilon + \theta, p_2 + 4\delta + 6\epsilon + \theta\}$	$T_{2,12} \geq T_{1,14}$
$C_{2,13}$	$T_{2,13} = \max\{5\delta + 6\epsilon + \theta, p_1 + 2\epsilon + \theta, p_2 + 3\delta + 5\epsilon + \theta\}$	$T_{2,13} \geq T_{3,4}$
$C_{2,14}$	$T_{2,14} = \max\{6\delta + 6\epsilon, p_1 + 2\delta + 3\epsilon, p_2 + 3\delta + 4\epsilon\}$	$T_{2,14} \geq T_{3,5}$
$C_{2,15}$	$T_{2,15} = T_{1,18}$	$T_{2,15} \geq T_{1,18}$
$C_{2,16}$	$T_{2,16} = T_{1,13}$	$T_{2,16} \geq T_{1,13}$

Table 4.3. Cycle-Time Expressions for 1-Unit Cycles $C_{2,k}$ for $m = 2$.

The cycle time t_d^m for C_d^m can easily be calculated as

$$t_d^m = (m+1)\delta + 2(m+1)\epsilon + m\theta + \sum_{k=1}^{m} w_k. \tag{4.1}$$

The time sequence corresponding to C_d^m can be represented as

$\{\epsilon,\ [\delta, w_1, \epsilon, \theta, \epsilon],\ \ldots\ [\delta(1), w_i, \epsilon, \theta, \epsilon(2)],\ \ldots\ [\delta, w_m, \epsilon, \theta, \epsilon],\ \delta, \epsilon\}$.

At point (1) above: the robot returns to unload M_i.

At point (2) above: the robot has just loaded a part on M_i.

From this time sequence, we can calculate w_i as

$$w_i = \max\{0, p_i - (m+1)\delta - 2m\epsilon - (m-1)\theta - \sum_{k \neq i} w_k\}. \tag{4.2}$$

Adding $-w_i + \sum w_k$ to both sides of (4.2) yields

$$\sum w_k = \max\{-w_i + \sum w_k, p_i - (m+1)\delta - 2m\epsilon - (m-1)\theta\}. \tag{4.3}$$

Cycle	Cycle Time	Dominance Relation
$C_{3,1}$	$T_{3,1} = \max\{5\delta + 6\epsilon, p_1 + 3\delta + 5\epsilon, p_2 + 3\delta + 5\epsilon\}$	$T_{3,1} \geq T_{1,1}$
$C_{3,2}$	$T_{3,2} = T_{1,13}$	$T_{3,2} \geq T_{1,13}$
$C_{3,3}$	$T_{3,3} = \max\{p_1 + 4\delta + 6\epsilon, p_2 + 2\delta + 4\epsilon\}$	undominated
$C_{3,4}$	$T_{3,4} = \max\{4\delta + 6\epsilon + \theta, p_1 + \theta + 2\epsilon, p_2 + 2\delta + 4\epsilon\}$	undominated
$C_{3,5}$	$T_{3,5} = \max\{6\delta + 6\epsilon, p_1 + 2\delta + 3\epsilon, p_2 + 2\delta + 3\epsilon\}$	undominated
$C_{3,6}$	$T_{3,6} = \max\{5\delta + 6\epsilon + \theta, p_1 + 2\epsilon + \theta, p_2 + 2\delta + 3\epsilon\}$	undominated
$C_{3,7}$	$T_{3,7} = \max\{p_1 + 5\delta + 6\epsilon, p_2 + 2\delta + 3\epsilon\}$	$T_{3,7} \geq T_{1,1}$
$C_{3,8}$	$T_{3,8} = T_{1,18}$	$T_{3,8} \geq T_{1,18}$
$C_{3,9}$	$T_{3,9} = \max\{5\delta + 6\epsilon + \theta, p_1 + 2\delta + 3\epsilon, p_2 + \theta + 2\epsilon\}$	undominated
$C_{3,10}$	$T_{3,10} = \max\{3\delta + 6\epsilon + 2\theta, p_1 + \theta + 2\epsilon, p_2 + \theta + 2\epsilon\}$	undominated
$C_{3,11}$	$T_{3,11} = \max\{p_1 + 3\delta + 6\epsilon + \theta, p_2 + 2\epsilon + \theta\}$	undominated
$C_{3,12}$	$T_{3,12} = T_{3,10} + \theta$	$T_{3,12} \geq T_{3,10}$
$C_{3,13}$	$T_{3,13} = T_{3,11} + \theta$	$T_{3,13} \geq T_{3,11}$
$C_{3,14}$	$T_{3,14} = \max\{5\delta + 6\epsilon + \theta, p_1 + 3\delta + 5\epsilon + \theta, p_2 + 2\epsilon + \theta\}$	$T_{3,14} \geq T_{3,9}$
$C_{3,15}$	$T_{3,15} = \max\{p_1 + 4\delta + 6\epsilon + \theta, p_2 + 2\epsilon + \theta\}$	$T_{3,15} \geq T_{3,11}$
$C_{3,16}$	$T_{3,16} = \max\{4\delta + 6\epsilon + \theta, p_1 + 2\delta + 4\epsilon, p_2 + 2\epsilon + \theta\}$	undominated

Table 4.4. Cycle-Time Expressions for 1-Unit Cycles $C_{3,k}$ for $m = 2$.

Cycle	Cycle Time	Dominance Relation
$C_{4,1}$	$T_{4,1} = \max\{6\delta + 6\epsilon, p_1 + 3\delta + 4\epsilon, p_2 + 3\delta + 4\epsilon\}$	$T_{4,1} = T_{1,18}$
$C_{4,2}$	$T_{4,2} = 3\delta + 6\epsilon + p_1 + p_2$	$T_{4,2} = T_{1,13}$

Table 4.5. Cycle-Time Expressions for 1-Unit Cycles $C_{4,k}$ for $m = 2$.

If $w_i > 0$, (4.3) implies that $\sum w_k = p_i - (m+1)\delta - 2m\epsilon - (m-1)\theta$. If $w_i = 0$, (4.3) implies that $\sum w_k \geq p_i - (m+1)\delta - 2m\epsilon - (m-1)\theta$. It follows that

$$\sum w_k = \max\{0, \max\{p_i\} - (m+1)\delta - 2m\epsilon - (m-1)\theta\}. \qquad (4.4)$$

Substituting (4.4) into (4.1) yields

$$t_d^m = \max\{(m+1)\delta + 2(m+1)\epsilon + m\theta, \max\{p_i\} + 2\epsilon + \theta\}. \qquad (4.5)$$

4.4 Dual-Gripper Cells with Small Gripper Switch Times

In most practical environments, the gripper switch time θ is typically less than the inter-machine travel times and the machine processing times. Thus, it is reasonable to assume that $\theta \leq \min\{\delta, p_1, \ldots, p_m\}$. This assumption holds in this section and Sections 4.5-4.7. Under this assumption, we now show that the dual-gripper robot cycle C_d^m is optimal for $RF_m^2|(free,A,cyclic\text{-}1)|\mu$.

THEOREM 4.1 *Assume $\theta \leq \min\{\delta, p_1, \ldots, p_m\}$. A lower bound for cycle times for 1-unit robot move cycles using either a single-gripper or dual-gripper robot is given by the right hand side of (4.5):*

$$LB = \max\{(m+1)\delta + 2(m+1)\epsilon + m\theta, \max\{p_i\} + 2\epsilon + \theta\}. \qquad (4.6)$$

Proof. In part A of the proof, we show that $\max\{p_i\} + 2\epsilon + \theta$ is a lower bound. In part B, we show that $(m+1)\delta + 2(m+1)\epsilon + m\theta$ is a lower bound.

Part A: Consider any machine $M_i, 1 \leq i \leq m$, and a schedule σ. Since σ is cyclic, we may assume it to be of the form $(M_i^-, \sigma_1, M_i^+, \sigma_2)$, where at least one of σ_1 and σ_2 is not empty. We consider two cases.

Case 1. $\sigma_2 \neq \emptyset$. The time from the start of loading a job on M_i until the completion of unloading on M_i is at least $p_i + 2\epsilon$ (covering the subschedule (M_i^-, σ_1, M_i^+). The robot will be engaged for at least an additional amount of time δ in order to complete σ_2. Thus, a lower bound on the schedule length will be $p_i + 2\epsilon + \delta \geq p_i + 2\epsilon + \theta$.

Case 2. $\sigma_1 \neq \emptyset, \sigma_2 = \emptyset$. This case can occur only with a dual-gripper robotic cell. The cycle may be denoted by (M_i^+, M_i^-, σ_1). The subcycle (M_i^+, M_i^-) requires $2\epsilon + \theta$ time units, and the delay from the completion of M_i^- to the start of M_i^+ is of length at least p_i, for a total time of $p_i + 2\epsilon + \theta$ time units.

Cases 1 and 2 prove that $\max\{p_i\} + 2\epsilon + \theta$ is a lower bound on the cycle times.

Part B: Recall that for any machine M_i, the schedule σ is of the form $(M_i^-, \sigma_1, M_i^+, \sigma_2)$, where at least one of σ_1 and σ_2 is not empty. We shall compute lower bounds for the aggregate residence times of the robot at

each machine and for the aggregate robot transportation time between machines. The sum of these lower bounds will give us the bound of $(m+1)\delta + 2(m+1)\epsilon + m\theta$. In our discussion, $p_0 = p_{m+1} = 0$ is the "processing time" at I/O. For residence times at the machines, there are four cases to consider.

Case 1. $M_i = M_0$ (the "machine" is I/O). The robot is occupied at M_0 for 2ϵ time units.

Case 2. $\sigma_2 = \emptyset$. The robot is occupied at M_i for at least $2\epsilon + \theta$ time units.

Case 3. $\sigma_1 = \emptyset$. The robot is occupied at M_i for at least $2\epsilon + p_i$ time units.

Case 4. $\sigma_1 \neq \emptyset \neq \sigma_2$. The robot is occupied at M_i for at least 2ϵ time units (split between two visits).

Assuming that the number of machines included in Cases 2, 3, and 4, respectively, are u, v, and $w = m - u - v$, and that V is the set of machines included in Case 3, we get an aggregate residence time for the robot at all machines to be

$$2\epsilon + u(2\epsilon + \theta) + v(2\epsilon) + \sum_{i \in V} p_i + w(2\epsilon) = 2(m+1)\epsilon + u\theta + \sum_{i \in V} p_i. \quad (4.7)$$

A robot movement from M_i to M_j occurs when an operation M_i^- or M_i^+ is followed immediately by a robot operation M_j^- or M_j^+, where $j \neq i$. Such a movement is "incident" to both M_i and M_j, and requires δ time units if M_i is adjacent to M_j, and a multiple of δ time units otherwise.

For each machine in each cycle, there is at least one movement into and one movement out of M_i, so M_i is incident to at least two movements. If M_i is included in Case 4, there are in each cycle at least two movements into and two movements out of M_i, so M_i will be incident to at least four movements. Thus, the aggregate of incidences over all machines is at least $2(1+u+v) + 4w = 2(m+1) + 2w$. Hence, the total number of movements, which is half the aggregate incidences, is at least $m+1+w$. These movements require a minimum aggregate time of $(m+1+w)\delta$. A lower bound on the cycle times is obtained by adding this to (4.7). Since $\theta \leq \min\{\delta, p_1, \ldots, p_m\}$, we obtain $(m+1)\delta + 2(m+1)\epsilon + m\theta$ as the lower bound. ∎

Since cycle C_d^m achieves this lower bound, the following corollary is immediate.

COROLLARY 4.1 *C_d^m is optimal among all single-gripper and dual-gripper 1-unit cyclic schedules under the assumption that $\theta \leq \min\{\delta, p_1, \ldots, p_m\}$.*

4.5 Comparing Dual-Gripper and Single-Gripper Cells

A fundamental question concerning dual-gripper cells is the extent of improvement in productivity that they offer as compared to single-gripper cells. In this section, we present an analysis for additive travel-time cells. Let T_s (resp., T_d) denote the optimal cycle time for problem $RF_m^{1,\circ}|(free,A,cyclic\text{-}1)|\mu$ (resp., $RF_m^{2,\circ}|(free,A,cyclic\text{-}1)|\mu$).

THEOREM 4.2 *For m-machines robotic cells, $T_s/T_d \leq 2$, and this bound is tight.*

Proof. We define an instance as a vector $I = (\delta, \epsilon, \theta, p_1, \ldots, p_m)$, where all parameters are non-negative. Let $T_s(I)$ and $T_d(I)$ denote the optimal values for the instance I in single-gripper and dual-gripper cells, respectively. Note that θ is not a parameter in the calculation of $T_s(I)$, while $T_d(I) = T_d(\delta, \epsilon, \theta, p_1, \ldots, p_m)$ is monotonically nondecreasing in θ. It follows that for purposes of the worst case analysis of T_s/T_d, we may assume $\theta = 0$.

We define the instance $I'' = (\delta, \epsilon, 0, p_1, ..., p_m)$ to be the same as instance I with the one exception that $\theta = 0$. Based on the above comments,

$$T_s(I)/T_d(I) \leq T_s(I'')/T_d(I'').$$

Since I'' has $\theta = 0$, it follows that $T_d(I'') = \max\{(m+1)\delta + 2(m+1)\epsilon, \max\{p_i\} + 2\epsilon\}$. Since $T_d(I'')$ depends on the processing times only through $\max\{p_i\}$, it follows that for the purpose of the worst case analysis, we may assume that all p_i's are equal: if we increase all p_i's to $\max\{p_i\}$, we would not increase the optimal cycle time for the dual-gripper problem, while the duration for the single-gripper problem may increase. Thus, we assume that $\theta = 0$ and $p_1 = ... = p_m \equiv p$, and define the corresponding instance $I' = (\delta, \epsilon, 0, p, ..., p)$. Clearly,

$$T_s(I)/T_d(I) \leq T_s(I'')/T_d(I'') \leq T_s(I')/T_d(I').$$

Thus, to prove the upper bound, we may restrict our attention to the instance I'. For the remainder of the proof, the notation T_s and T_d refer to $T_s(I')$ and $T_d(I')$, respectively.

Next, we define a 1-unit cyclic schedule σ for a single-gripper robot so that the cycle time T_σ is at most twice the optimal dual-gripper cycle time T_d. The single-gripper cycle σ starts with the robot empty at I/O and with a job in progress at each of the even-numbered machines. In each cycle, the robot makes two visits to each of the machines. On its first visit, each even-numbered machine is unloaded and each odd-numbered machine is loaded. During its second visit, each odd-numbered machine is unloaded and each even-numbered machine is loaded. The analysis for m odd is slightly different from that for m even. We deal first with the case when m is even.

To begin with, assume that part P_i is on M_m, P_{i+1} is on M_{m-2}, ..., and $P_{i+m/2-1}$ is on M_2. To help clarify the proof, we provide in Table 4.6 an example of the sequence of operations and their durations for cycle σ in a six-machine cell. For simplicity and without loss of generality, we let $i = 1$, i.e., the initial parts on M_6, M_4, and M_2 are, respectively, P_1, P_2, and P_3. Note that the operation *wait* at I/O (the last operation shown in the table) is a dummy operation with $w_0 = 0$, and is included to help the reader appreciate the symmetry associated with the schedule. The order of the operations in Table 4.6 is as follows: the 12 operations in Column 1 followed by the 12 operations in Column 3 and, finally, the 11 operations in Column 5. It is easy to see that the subsequence ($P_j \rightarrow R$, $R \rightarrow M_k$, $P_j \rightarrow M_k$, $R \rightarrow M_{k+1}$, *wait*) is repeated seven times, for appropriate values of j and k, with the last wait (at I/O) in the cycle being the dummy operation mentioned above. Correspondingly, we have a sequence of durations that repeats seven times the subsequence (ϵ, δ, ϵ, δ, w_i) for appropriate values of i. The w_i's are derived as described in Chapter 3. We have

$$w_i = \max\left\{0, p - 7\delta - 6\epsilon - \sum_{j=1}^{\lfloor (7-i)/2 \rfloor} w_{i-1+2j} - \sum_{j=1}^{\lfloor (i-1)/2 \rfloor} w_{i-2j}\right\}, \quad for \ 1 \le i \le 6.$$

Operation (1)	Duration (2)	Operation (3)	Duration (4)	Operation (5)	Duration (6)
$P_4 \to R$	ϵ	$P_2 \to M_5$	ϵ	$wait$	w_3
$R \to M_1$	δ	$R \to M_6$	δ	$P_3 \to R$	ϵ
$P_4 \to M_1$	ϵ	$wait$	w_6	$R \to M_4$	δ
$R \to M_2$	δ	$P_1 \to R$	ϵ	$P_3 \to M_4$	ϵ
$wait$	w_2	$R \to I/O$	δ	$R \to M_5$	δ
$P_3 \to R$	ϵ	$P_1 \to I/O$	ϵ	$wait$	w_5
$R \to M_3$	δ	$R \to M_1$	δ	$P_2 \to R$	ϵ
$P_3 \to M_3$	ϵ	$wait$	w_1	$R \to M_6$	δ
$R \to M_4$	δ	$P_4 \to R$	ϵ	$P_2 \to M_6$	ϵ
$wait$	w_4	$R \to M_2$	δ	$R \to I/O$	δ
$P_2 \to R$	ϵ	$P_4 \to M_2$	ϵ	$wait$	$w_0 = 0$
$R \to M_5$	δ	$R \to M_3$	δ	$---$	$---$

Table 4.6. A 1-Unit Cycle for $m = 6$.

The cycle time T_σ for this particular cycle is given by $T_\sigma = 14\delta + 14\epsilon + \sum w_i$. A solution is given by

$$\begin{aligned} w_1 &= 0, \\ w_i &= \max\{0, (p - 7\delta - 6\epsilon)/3\}, \text{ for } 2 \le i \le 6. \end{aligned}$$

Generalizing the ideas illustrated in the example, we will now present the case in which m is an arbitrary even number.

For m an arbitrary even number, the cycle we get is given by $(m+1)$ copies of the subsequence ($P_j \to R$, $R \to M_k$, $P_j \to M_k$, $R \to M_{k+1}$, $wait$) for appropriate values of j and k with the last wait being the dummy wait at I/O. Correspondingly, we have a sequence of durations which repeats $(m+1)$ times the subsequence (ϵ, δ, ϵ, δ, w_i) for appropriate values of i. The w_i's appear in the order $(w_2, w_4, \ldots, w_m, w_1, w_3, \ldots, w_{m-1})$. The w_i's are calculated in the usual fashion. For $1 \le i \le m$, we may write

$$w_i = \max\{0, p - (m+1)\delta - m\epsilon - \sum_{j=1}^{\lfloor (m-i+1)/2 \rfloor} w_{i-1+2j} - \sum_{j=1}^{\lfloor (i-1)/2 \rfloor} w_{i-2j}\}.$$

The cycle time T_σ for this particular cycle is given by $T_\sigma = 2(m+1)\delta + 2(m+1)\epsilon + \sum w_i$. A solution is given by

$$\begin{aligned} w_1 &= 0, \\ w_i &= \max\{0, [p-(m+1)\delta - m\epsilon]/[m/2]\}, \text{ for } 2 \le i \le m. \end{aligned}$$

We may now calculate T_σ for even values of m. Observe that

$$T_\sigma = 2(m+1)\delta + 2(m+1)\epsilon + \sum w_i.$$

Thus,

$$T_\sigma = \begin{cases} 2(m+1)\delta + 2(m+1)\epsilon, & \text{if } p-(m+1)\delta - m\epsilon \le 0, \\ (1/m)[2(m+1)\delta + 4m\epsilon + 2(m-1)p], & \text{otherwise.} \end{cases}$$

Note that T_σ may not be an optimal 1-unit cycle for the single-gripper robotic cell. Since $T_s \le T_\sigma$, it is sufficient to establish the worst case ratio for T_σ/T_d. Keeping in mind that $T_d = \max\{(m+1)\delta + 2(m+1)\epsilon, p+2\epsilon\}$, we compute the worst case ratio:

- For $p-(m+1)\delta - m\epsilon \le 0$, we have

$$\begin{aligned} T_\sigma/T_d &= [2(m+1)(\delta+\epsilon)]/\max\{(m+1)(\delta+2\epsilon), p+2\epsilon\}, \\ &\le [2(m+1)(\delta+\epsilon)]/[(m+1)(\delta+2\epsilon)] \le 2. \end{aligned}$$

- For $p-(m+1)\delta - m\epsilon > 0$, we have

$$\begin{aligned} T_\sigma/T_d &= (1/m)[2(m+1)\delta + 4m\epsilon + 2(m-1)p]/\max\{(m+1)\delta \\ &\quad + 2(m+1)\epsilon, p+2\epsilon\}, \\ &\le [(1/m)(2(m+1)\delta + 4m\epsilon)]/[(m+1)\delta + 2(m+1)\epsilon] \\ &\quad + [(1/m)(2(m-1)p)]/[p+2\epsilon], \\ &\le (1/m)(2) + (1/m)(2(m-1)) = 2. \end{aligned}$$

This completes the proof that $T_\sigma/T_d \le 2$ for even values of m.

We now prove the result for odd values of m. For illustration, we specify in Table 4.7 a schedule σ for $m = 5$ which, as before, starts with the robot empty at I/O and the machines M_2 and M_4 occupied. Without loss of generality, part P_1 may be assumed to be on M_4. For an arbitrary odd number m, the cycle σ obtained is described as follows:

Operation (1)	Duration (2)	Operation (3)	Duration (4)	Operation (5)	Duration (6)
$P_3 \to R$	ϵ	$P_1 \to R$	ϵ	$P_2 \to R$	ϵ
$R \to M_1$	δ	$R \to M_5$	δ	$R \to M_4$	δ
$P_3 \to M_1$	ϵ	$P_1 \to M_5$	ϵ	$P_2 \to M_4$	ϵ
$R \to M_2$	δ	$R \to M_1$	2δ	$R \to M_5$	δ
$wait$	w_2	$wait$	w_1	$wait$	w_5
$P_2 \to R$	ϵ	$P_3 \to R$	ϵ	$P_1 \to R$	ϵ
$R \to M_3$	δ	$R \to M_2$	δ	$R \to I/O$	δ
$P_2 \to M_3$	ϵ	$P_3 \to M_2$	ϵ	$P_1 \to I/O$	ϵ
$R \to M_4$	δ	$R \to M_3$	δ	$- - -$	$- - -$
$wait$	w_4	$wait$	w_3	$- - -$	$- - -$

Table 4.7. A 1-Unit Cycle for $m = 5$.

$(m-1)/2$ copies of the subsequence ($P_i \to R$, $R \to M_k$, $P_i \to M_k$, $R \to M_{k+1}$, $wait$) for appropriate values of i and k, followed by one copy of ($P_1 \to R$, $R \to M_m$, $P_1 \to M_m$, $R \to M_1$, $wait$), followed by $(m-1)/2$ copies of the subsequence ($P_i \to R$, $R \to M_k$, $P_i \to M_k$, $R \to M_{k+1}$, $wait$) for appropriate values of i and k, followed by one copy of ($P_1 \to R$, $R \to I/O$, $P_1 \to I/O$).

The w_i's are derived as described in Chapter 3. We have

$$w_i = \max\{0, p-(m+1)\delta-(m-2\lceil i/2\rceil+2\lfloor i/2\rfloor+1)\epsilon - \sum_{j=1}^{\lfloor (m-i+1)/2\rfloor} w_{i-1+2j} - \sum_{j=1}^{\lfloor (i-1)/2\rfloor} w_{i-2j}\}.$$

Note that $(m-2\lceil i/2\rceil+2\lfloor i/2\rfloor+1)\epsilon$ equals $(m+1)\epsilon$ when i is even, and equals $(m-1)\epsilon$ when i is odd. The cycle time is $T_\sigma = 2(m+1)\delta + 2(m+1)\epsilon + \sum w_i$. We consider three cases:

- Case 1. $p-(m+1)\delta-(m+1)\epsilon \le p-(m+1)\delta-(m-1)\epsilon \le 0$.

  In this case, $w_1 = \ldots = w_m = 0$, $T_\sigma = 2(m+1)\delta + 2(m+1)\epsilon$, and $T_\sigma/T_d \le [2(m+1)\delta + 2(m+1)\epsilon]/[(m+1)\delta + 2(m+1)\epsilon] \le 2$.

- Case 2. $p-(m+1)\delta-(m+1)\epsilon \le 0 < p-(m+1)\delta-(m-1)\epsilon$.

In this case, $w_1 = p - (m+1)\delta - (m-1)\epsilon$, $w_2 = \ldots = w_m = 0$, and $T_\sigma = (m+1)\delta + (m+1)\epsilon + 2\epsilon + p$. Therefore, $T_\sigma/T_d \leq [(m+1)\delta + (m+1)\epsilon]/[(m+1)\delta + 2(m+1)\epsilon] + (p+2\epsilon)/(p+2\epsilon) \leq 1 + 1 = 2$.

- Case 3. $0 < p - (m+1)\delta - (m+1)\epsilon \leq p - (m+1)\delta - (m-1)\epsilon$.

  In this case, we have
  $w_1 = [p - (m+1)\delta - (m+1)\epsilon]/[(m+1)/2] + 2\epsilon$,
  $w_2 = \ldots = w_m = [p - (m+1)\delta - (m+1)\epsilon]/[(m+1)/2] = w_1 - 2\epsilon$,
  $T_\sigma = 2\delta + 4\epsilon + (2mp)/(m+1)$.
  Therefore, $T_\sigma/T_d \leq [2\delta + 4\epsilon]/[(m+1)\delta + 2(m+1)\epsilon] + [(2mp)/(m+1)]/[p + 2\epsilon]$. Thus, $T_\sigma/T_d \leq [2/(m+1)] + 2m/(m+1) = 2$.

This completes the proof that 2 is an upper bound on T_σ/T_d for instances of the form of I' with $\theta = 0$ and $p_1 = ... = p_m \equiv p$. As discussed at the beginning of the proof, the bound holds for all instances.

To show that 2 is a tight bound on T_s/T_d, we use the following data: $(m+1)\delta = p_1 = \ldots = p_m = p > 0$ and $\epsilon = \theta = 0$. In this case, $T_d = (m+1)\delta$. The cycle time for the schedule σ is $T_\sigma = 2(m+1)\delta$. For any other single-gripper schedule τ for this instance, we show that $T_\tau \geq 2(m+1)\delta$. The proof holds for both odd and even values of m.

The schedules with which we are concerned may be described as cyclic permutations of the sub-schedules (M_{j-1}^+, M_j^-), $1 \leq j \leq m+1$. Each such cyclic permutation yields a 1-unit cyclic schedule, and every 1-unit cyclic schedule can be so described. Assume τ to be a schedule of duration less than $2(m+1)\delta$. We show that such a schedule cannot exist. A few observations about τ are in order:

1. Corresponding to each (M_{j-1}^+, M_j^-), $1 \leq j \leq m+1$, there is a travel time of δ between M_{j-1} and M_j. Thus, the robot is occupied for $(m+1)\delta$ units of time for these moves.

2. The sub-sequences (M_j^-, M_{j-1}^+), $1 \leq j \leq m+1$, cannot occur. This follows from the fact that the sub-sequences (M_{j-1}^+, M_j^-), $1 \leq j \leq m+1$, do necessarily occur.

3. For $1 \leq j \leq m$, if the sub-schedule (M_j^-, M_j^+) is part of a schedule, then the duration of the schedule is at least $2(m+1)\delta$. This results from the $(m+1)\delta$ units of robot travel time mentioned in observation (1) above, and a robot waiting time of $(m+1)\delta$ at M_j.

4. It follows that for the purpose of finding a schedule τ of duration less than $(m+1)\delta$, we may assume, for $1 \leq j \leq m$, that M_j^+ is immediately preceded by $M_{i(j)}^-$ for some $i(j) \neq j$. Hence, a travel time of *at least* δ is associated with the sub-schedule $(M_{i(j)}, M_j)$. It follows that in addition to the travel time of $(m+1)\delta$ mentioned in observation (1), there must be an additional travel time of *at least* $m\delta$ associated with $(M_{i(j)}, M_j)$ for $j \neq 0$, for a total travel duration of *at least* $2(m+1)\delta - \delta$. Thus, there cannot exist an additional travel time associated with τ other than that just mentioned. In particular, this implies that (M_0^-, M_0^+) must be part of the schedule τ.

We now pose the question: For which value of h is (M_m^-, M_h^+) a sub-schedule of τ? By observation (2), $h \neq m-1$. By observation (3), $h \neq m$. By observation (4), $h \neq 0$, or equivalently $h \neq m+1$, since (M_0^-, M_0^+) is a sub-schedule. It follows that regardless of what h is, there is a travel time to M_h of *at least* 2δ associated with (M_m^-, M_h^+). Thus, by observation (1) there is an aggregate travel time of $(m+1)\delta$ associated with (M_{j-1}^+, M_j^-), $1 \leq j \leq m+1$. By observation (4), there is an additional travel time of at least $(m-1)\delta$ associated with each $(M_{i(j)}, M_j)$ for $j \neq 0, h$. We have just seen that associated with (M_m^-, M_h^+), there is a travel time of at least 2δ. Hence, the total travel time of τ is at least $2(m+1)\delta$. This completes the proof. ∎

4.6 Comparison of Productivity: Computational Results

One obvious approach to assess the productivity advantage of a dual-gripper cell over a single-gripper cell is to estimate the performance ratio of the cycle time of the best single-gripper cycle to that of the best dual-gripper cycle. In general, one has to consider all possible multi-unit cycles to arrive at the optimal value of this ratio. For large m, explicit enumeration becomes impractical even if we limit our search to all 1-unit cycles. Drobouchevitch et al. [50] show that for single part-type production in a dual-gripper cell, the increase in the number of machines in the cell leads to an increase in the number of possible 1-unit cycles that is much higher as compared to that in a single-gripper cell. For example, for $m = 10$, there are 642,787,488,000 feasible 1-unit cycles for a dual-gripper cell (Table 4.8) and $10! = 3,628,800$ feasi-

ble 1-unit cycles for a single-gripper cell. We describe below a simple heuristic, ProdRatio, to estimate the ratio T_s/T_d for the class of 1-unit cycles. Recall that T_s is the optimal single-gripper cycle time for problem $RF_m^1|(free,A,cyclic\text{-}1)|\mu$, and T_d is the optimal dual-gripper cycle time for problem $RF_m^2|(free,A,cyclic\text{-}1)|\mu$.

Number of machines m	Number of 1-unit cycles (Dual-gripper cell) Drobouchevitch et al. [50]	Number of 1-unit cycles, $m!$ (Single-gripper cell) Sethi et al. [142]
1	6	1
2	46	2
3	456	6
4	5 688	24
5	86 640	120
6	1 568 880	720
7	33 022 080	5 040
8	793 215 360	40 320
9	2 1 423 7 09 440	362 880
10	64 2 787 4 88 000	3 628 800

Table 4.8. Number of 1-Unit Cycles for both Dual-Gripper and Single-Gripper m-Machine Robotic Cells.

For an instance, algorithm ProdRatio computes cycle time estimates by exploring 1-unit cycles for both single-gripper and dual-gripper cells, and then finds the performance ratio based upon these estimates. Let $\hat{T}_s$ and $\hat{T}_d$ denote the estimates for T_s and T_d, respectively.

Algorithm ProdRatio

Input: An instance for both $RF_m^1|(free,A,cyclic\text{-}1)|\mu$ and for $RF_m^2|(free, A,cyclic\text{-}1)|\mu$ is specified by m, δ, ϵ, $p_1,\ldots,p_m$, and θ.

Step 1: Find the minimum cycle time (t_{min}) among all $m!$ 1-unit cycles for the single-gripper problem $RF_m^1|(free,A,cyclic\text{-}1)|\mu$. Set $\hat{T}_s = t_{min}$.

Step 2:

2a. If $m = 2$, then find the minimum cycle time (T_{min}) among all undominated 13 1-unit cycles – $C_{1,1}$,$C_{1,4}$, $C_{1,13}$,$C_{1,14}$,$C_{1,15}$,$C_{3,3}$,$C_{3,4}$,$C_{3,5}$,

$C_{3,6}$,$C_{3,9}$,$C_{3,10}$,$C_{3,11}$,$C_{3,16}$ – for $RF_2^2|(free,A,cyclic\text{-}1)|\mu$. Set $\hat{T}_d = T_{min}$.

2b. If $m > 2$, then find the cycle time (t_d^m) for the 1-unit cycle C_d^m for $RF_m^2|(free,A,cyclic\text{-}1)|\mu$. Set $\hat{T}_d = t_d^m = \max\{(m+1)\delta + 2(m+1)\epsilon + m\theta, \max\{p_j\} + 2\epsilon + \theta\}$.

Step 3: Find the ratio $\hat{T}_s/\hat{T}_d$.

Step 4: Output: $\hat{T}_s$, $\hat{T}_d$ and $\hat{T}_s/\hat{T}_d$. Terminate.

Theorems 4.3 and 4.4 describe the performance of ProdRatio. Based on our discussion so far, the following results are immediate.

THEOREM 4.3 *For $m = 2$, ProdRatio yields a performance ratio with the value T_s/T_d.*

THEOREM 4.4 *Given any instance of an m-machine problem satisfying the inequality $\theta \leq \min\{\delta, p_1, \ldots, p_m\}$, ProdRatio yields a performance ratio with value T_s/T_d.*

Theorem 4.4 leaves open the question of quality of the ratio $\hat{T}_s/\hat{T}_d$ yielded by ProdRatio for the case $\theta > \min\{\delta, p_1, \ldots, p_m\}$. To address this case, we describe a simulation that will bound the ratio $\hat{T}_s/\hat{T}_d$ produced by the algorithm. The simulation works as follows: generate an instance satisfying $\theta > \min\{\delta, p_1, \ldots, p_m\}$, let $\theta' = \min\{\delta, p_1, \ldots, p_m\}$, and execute ProdRatio on the instance obtained by substituting θ' for θ. Let $r = \hat{T}_s/\hat{T}_d$ denote the ratio produced by ProdRatio applied to the original instance, and let $r' = \hat{T}_s/\hat{T}'_d$ denote the ratio produced by the algorithm applied to the transformed instance that uses θ' instead of θ. Note that $\hat{T}'_d = \max\{(m+1)\delta + 2(m+1)\epsilon + m\theta', \max\{p_j\} + 2\epsilon + \theta'\}$. We have $\hat{T}_s/\hat{T}_d \leq \hat{T}_s/\hat{T}'_d$. Theorem 4.4 applies to the transformed problem. Thus, the maximum percentage by which r may underestimate the ratio $\hat{T}_s/\hat{T}_d$ is given by $[100(r' - r)]/r$.

Next, we provide the computational results from applying heuristic ProdRatio for instances generated for the following two cases. *Case 1*: $\theta \leq \min\{\delta, p_1, \ldots, p_m\}$ and *Case 2*: $\theta > \min\{\delta, p_1, \ldots, p_m\}$. Note that for linear and semicircular layouts, the best 1-unit single-gripper cycle can be found in polynomial time in m (Chapter 3, Crama and van de Klundert [40]). This result, however, does not hold for the circular layout

where the input and output devices are located in one place. Since we do not know of any existing efficient algorithm for a circular layout, we let Step 1 of ProdRatio enumerate the $m!$ cycles. For $m \leq 12$, we use exhaustive search to find an optimal 1-unit cycle as this search is computationally feasible. For $m \geq 13$, we apply a heuristic based on a Genetic Algorithm (GA) to find a good 1-unit cycle.

Genetic Algorithms belong to the class of heuristic optimization techniques that utilize randomization as well as directed smart search to seek a global optimum. The creation of GAs was inspired by evolutionary processes through which life is believed to have evolved in its present forms (Goldberg [68]). When applied to the robotic cell scheduling problem, a GA views a sequence of machine loadings by the robot as an individual candidate sequence or solution. For example, for $m = 4$, the candidate solution $\pi = (5, 4, 2, 3, 1)$ denotes the sequence of machine loadings in the associated 1-unit cycle: $(M_5^-, M_4^-, M_3^-, M_2^-, M_1^-)$ (see Chapter 3). A number of such individual solutions constitutes a population. Each individual solution is characterized by its fitness (e.g., cycle time value). The GA works iteratively with the members of the population using operators such as *crossover* and *mutation*; each new iteration is referred to as a *generation*. For details, we refer the reader to Murata and Ishibuchi [123]. A key challenge in the application of GAs is the optimization of the computational effort in balancing exploration of the solution space and exploitation of the features of good solutions or sequences produced along the way. This balance is affected greatly by the choices of the different GA parameters, including elite fraction (ef), population size (Ps), the probability of crossing two parents (Pc), the probability of mutation (Pm), and the number of generations (n_{gen}) (for details, we refer the reader to Goldberg [68], Davis [44]). After performing some trial runs, we found that the following values for these parameters are suitable for our experiments: $Ps = 100$, $Pc = 0.95$, $Pm = 0.1$, $ef = 0.5$, and $n_{gen} = 100$.

To test algorithm ProdRatio, we use data guided primarily by practical relevance. We generate three types of processing times of the parts: I_1, I_2, I_3. Under I_1 (resp., I_2 and I_3), the processing times are chosen randomly from $U[1, 15]$ (resp., $U[1, 25]$ and $U[1, 50]$). In practice, the value of θ is smaller than the processing times of the parts. We

use three values of θ: 0.25, 1.0, and 1.5. For all instances, we set $\epsilon = 0.25$ and $\delta = 1.0$. For each setting of the parameters, we generate five instances. Let r_k and $r_k^{'}$ denote, respectively, the ratio obtained by the heuristic and the upper bound for the kth instance of each setting. Let $\bar{r} = \sum_{k=1}^{5} r_k/5$, $\bar{r'} = \sum_{k=1}^{5} r'_k/5$, $r_{min} = \min_{1\leq k\leq 5}\{r_k\}$, $r_{max} = \max_{1\leq k\leq 5}\{r_k\}$, $e_k = 100(r_k^{'} - r_k)/r_k$, $e = \sum_{k=1}^{5} e_k/5$, $\ell = \min_{1\leq k\leq 5}\{e_k\}$, and $u = \max_{1\leq k\leq 5}\{e_k\}$.

m	Instance	θ	r_{min}	$\bar{r}$	r_{max}
2	I_1	0.25	1.2549	1.3330	1.5000
2	I_2	0.25	1.1206	1.1410	1.1646
2	I_3	0.25	1.0726	1.1384	1.3023
5	I_1	0.25	1.4237	1.5083	1.6098
5	I_2	0.25	1.1717	1.2120	1.2881
5	I_3	0.25	1.0854	1.1086	1.1475
10	I_1	0.25	1.6053	1.7105	1.8684
10	I_2	0.25	1.5152	1.7237	1.9211
10	I_3	0.25	1.0929	1.1281	1.2205
15	I_1	0.25	1.7297	1.8667	1.9820
15	I_2	0.25	1.8739	2.1514	2.2703
15	I_3	0.25	1.4426	1.4921	1.5276

Table 4.9. Performance Evaluation of Algorithm ProdRatio for Case 1 with $\theta = 0.25$.

Table 4.9 provides the performance ratios of ProdRatio for instances under Case 1 with $\theta = 0.25$. Each row of the table summarizes the results of the corresponding five instances. The fourth, fifth, and sixth columns show the minimum, mean, and the maximum performance ratios, respectively. The results for $\bar{r}$ show that, on average, the productivity improvements obtained by using a dual-gripper cell instead of a single-gripper cell range from 10% to 115%. The results also indicate that the productivity improvements are typically higher in cells with more machines. Over all the 60 instances, the smallest and the largest performance ratios are 1.0726 and 2.2703, respectively. Note, however, that the performance ratios r_k are an *overestimate* of the true productivity improvements: the heuristic used in Step 1 typically overestimates the single-gripper cycle time. A case in point is Row 11 of Table 4.9, where the ratios exceed the proven bound of 2 (Theorem 4.2) for some

m	Instance	θ	r_{min}	$\bar{r}$	r_{max}
2	I_1	1.0	1.1724	1.1950	1.2174
2	I_2	1.0	1.0980	1.1252	1.1613
2	I_3	1.0	1.0495	1.0597	1.0704
5	I_1	1.0	1.2414	1.3268	1.3929
5	I_2	1.0	1.1429	1.1744	1.2727
5	I_3	1.0	1.0722	1.0902	1.1373
10	I_1	1.0	1.0755	1.2227	1.3396
10	I_2	1.0	1.3585	1.4264	1.4906
10	I_3	1.0	1.0693	1.1302	1.3093
15	I_1	1.0	1.1795	1.3205	1.4872
15	I_2	1.0	1.3590	1.5000	1.5897
15	I_3	1.0	1.3366	1.3831	1.4588

Table 4.10. Performance Evaluation of Algorithm ProdRatio for Case 1 with $\theta = 1.0$.

m	I_i	θ	r_{min}	$\bar{r}$	r_{max}	$\bar{r}'$	ℓ	e	u
2	I_1	1.5	1.3333	1.1831	1.2857	-	-	-	-
2	I_2	1.5	1.0800	1.0982	1.1250	-	-	-	-
2	I_3	1.5	1.0435	1.0664	1.1053	-	-	-	-
5	I_1	1.5	1.2121	1.2606	1.3030	1.3703	6.45	8.73	17.86
5	I_2	1.5	1.1250	1.1952	1.2500	1.2235	2.04	2.37	3.03
5	I_3	1.5	1.0652	1.0955	1.1818	1.1117	1.10	1.46	2.33
10	I_1	1.5	1.0159	1.1079	1.1746	1.3170	18.87	18.87	18.87
10	I_2	1.5	1.1270	1.2191	1.2857	1.4491	18.87	18.87	18.87
10	I_3	1.5	1.0600	1.1352	1.2292	1.1472	1.01	1.05	1.12
15	I_1	1.5	1.0108	1.1269	1.2258	1.3436	19.23	19.23	19.23
15	I_2	1.5	1.0753	1.1807	1.3979	1.4077	19.23	19.23	19.23
15	I_3	1.5	1.3300	1.4058	1.4946	1.4702	0.99	4.39	12.05

Table 4.11. Performance Evaluation of Algorithm ProdRatio for Case 2 with $\theta = 1.5$.

instances. Table 4.10 reports the results for instances under Case 1 with $\theta = 1.0$. Table 4.11 reports the results for instances under Case 2 with $\theta = 1.5$, and also compares the performance ratios obtained by ProdRatio with an upper bound. As before, each row in Table 4.11 corresponds to the results of the corresponding five instances; the fourth, fifth, and sixth columns show the minimum, mean, and the maximum performance

ratio, respectively. Columns eight, nine, and ten show, respectively, the minimum, mean, and the maximum percentage deviation from the upper bound. The results for $\bar{r}$ show that productivity improvements averaging between 6% to 40% may be obtained by using a dual-gripper robot instead of a single-gripper robot. In general, the productivity improvements are higher in cells with more machines for a given type (i.e., I_1, I_2, or I_3) of processing times. The smallest and the largest performance ratio observed are 1.0108 and 1.4946, respectively, over all 60 instances reported in Table 4.11. On average, the performance of ProdRatio on smaller cells compares well with the upper bound. The mean relative deviations are all less than 8.73%, except for four cases where this value is either 18.87% or 19.23%. The smallest and the largest relative deviations are 0.0% and 19.23%, respectively. The results for $\bar{r}$ in Tables 4.9, 4.10, and 4.11 indicate that the productivity improvements decrease with an increase in the the value of the gripper switch time θ.

4.7 Efficiently Solvable Cases

This section identifies some efficiently solvable cases of $RF_m^{2,\circ}|(\textit{free,A, cyclic-1})|\mu$ in addition to the one in Corollary 4.1. Surprisingly, the $\theta \leq \delta$ assumption drastically simplifies the problem in the case of 1-unit cycles. Drobouchevitch et al. [50] show that problem $RF_m^2|(\textit{free,A,cyclic-1})|\mu$ with $\theta \leq \delta$ is efficiently solvable. A lower bound on an optimal solution is mentioned below. Let

$$D_\theta = \{k | p_k \leq \theta, 1 \leq k \leq m\}, \quad D_\theta^c = M \setminus D_\theta, \quad r = |D_\theta^c|. \tag{4.8}$$

LEMMA 4.1 *For problem $RF_m^2|(\textit{free,A,cyclic-1})|\mu$ with $\theta \leq \delta$, a lower bound on the length $T(C)$ of any 1-unit cycle C is:*

$$T(C) \geq \max \left\{ \max_{1 \leq k \leq m} p_k + \theta + 2\epsilon, \sum_{k \in D_\theta} p_k + r\theta + (m+1)(\delta + 2\epsilon) \right\}. \tag{4.9}$$

Proof. Similar to that for Corollary 4.1. See Drobouchevitch et al. [50] for details. ∎

We are now ready to demonstrate that problem $RF_m^2|(\textit{free,A,cyclic-1})|\mu$ is efficiently solvable provided $\theta \leq \delta$. The optimality of the 1-unit

cycle we obtain is guaranteed by its cycle length being equal to the lower bound (4.9).

Algorithm OPT1

Input: An instance of $RF_m^2|(free,A,cyclic\text{-}1)|\mu$. The sets D_θ and D_θ^c are as defined in (4.8).

Output: A 1-unit cycle C^*.

Initial setting: Machines M_k, $k \in D_\theta^c$, are occupied with a part; machines M_k, $k \in D_\theta$, are empty. The robot is positioned at I/O; both robot grippers are empty.

Step 1: The robot picks up a part from I/O.

Step 2: For k from 1 to m:

Step 2.1: The robot moves to machine M_k.

Step 2.2: If $k \in D_\theta^c$, then

- if necessary, the robot waits for w_k time units for a part on M_k to complete processing;
- the robot unloads a part from M_k (ϵ);
- the robot switches grippers (θ);
- the robot loads a part onto M_k (ϵ).

Step 2.3: Otherwise (i.e., $k \in D_\theta$),

- the robot loads a part onto M_k (ϵ);
- the robot waits for p_k time units for a part on M_k to be processed;
- the robot unloads a part from M_k (ϵ).

Step 3: The robot moves to the I/O hopper and drops a part at the output device of the I/O station.

It is easy to verify that the running time of algorithm OPT1 is $O(m)$. The theorem below analyzes its performance.

THEOREM 4.5 *For an instance of $RF_m^2|(free,A,cyclic\text{-}1)|\mu$ with $\theta \leq \delta$, OPT1 produces an optimal 1-unit cycle C^*. The cycle time of C^* is*

$$T(C^*) = \max \left\{ \max_{k \in D_\theta^c} p_k + 2\epsilon + \theta; \sum_{k \in D_\theta} p_k + r\theta + (m+1)(\delta + 2\epsilon) \right\}. \tag{4.10}$$

Proof. We leave it to the reader to verify the feasibility of cycle C^*. To prove the optimality of C^*, we show that its cycle time indeed achieves the lower bound represented by the expression on the right-hand side of inequality (4.9).

We consider two possible scenarios. First, suppose that there exists at least one $k \in D_\theta^c$ for which $w_k > 0$. Then, it is easy to see that

$$T(C^*) = p_k + 2\epsilon + \theta. \tag{4.11}$$

Now, suppose there is no positive w_k. Then,

$$T(C^*) = \sum_{k \in D_\theta} p_k + r\theta + (m+1)\delta + 2(m+1)\epsilon. \tag{4.12}$$

Combining (4.11) and (4.12) together, we obtain (4.10). By Lemma 4.1, the optimality of C^* follows. ∎

The result of Lemma 4.1 can be extended to the general case of $RF_m^2|(free,A,cyclic\text{-}k)|\mu$ (i.e., when the search for an optimal cyclic solution is not restricted to the class of 1-unit cycles) if we impose an assumption on the values of part processing times. The proof of the following theorem is similar to that of a result – Theorem 4.8 in Section 4.9 – we present later. We, therefore, state it here without a proof.

LEMMA 4.2 *For problem $RF_m^2|(free,A,cyclic\text{-}k)|\mu$ with $\theta \leq \delta$ and $\max_{1 \leq i \leq m}\{p_i\} \geq \delta$, a lower bound on the per unit cycle time of any k-unit cycle C is:*

$$\frac{T(C)}{k} \geq \max \left\{ \max_{1 \leq i \leq m} p_i + \theta + 2\epsilon; m\theta + (m+1)\delta + 2(m+1)\epsilon \right\}. \tag{4.13}$$

COROLLARY 4.2 *For problem $RF_m^2|(free,A,cyclic\text{-}k)|\mu$ with $\theta \leq \delta$ and $\max_{1 \leq i \leq m}\{p_i\} \geq \delta$, algorithm OPT1 obtains an optimal solution.*

4.8 Single-Gripper Cells with Output Buffers at Machines

In this section, we consider a model that is closely related to the dual-gripper problems we have studied so far. Specifically, we discuss a robotic cell served by a single-gripper robot that allows for temporary storage of processed parts at each machine. A unit-capacity buffer at a machine can be viewed as an alternative to an additional gripper. A local material handling device at a machine can move a completed part to the output buffer of that machine, without using the robot. Our aim is to investigate this model with machine buffers and compare it to a dual-gripper robotic cell. The analysis we present is from Drobouchevitch et al. [50]. We start with the formal definition of the problem.

Following the notation of our classification scheme in Chapter 2, we are interested in $RF^{1,\circ}_{m,\bar{1}}|(\textit{free,A,cyclic-1})|\mu$, the problem of obtaining a 1-unit cycle that maximizes throughput in an m-machine single-gripper robotic cell producing identical parts with a circular layout and with a unit-capacity output buffer at each machine; $\bar{1} = (1, 1, \ldots, 1)$ denotes that each machine has an output buffer of unit capacity. The processing requirements are the same as that for $RF^{2,\circ}_{m}|(\textit{free,A,cyclic-1})|\mu$. We denote the buffer at machine M_i by B_i. The pair (M_i, B_i) is referred to as the production unit Z_i.

The main purpose of an output buffer at a machine is to allow for temporary storage of a part that has been processed on the machine. This allows a single-gripper robot to unload and reload M_i during a single visit. From an optimization aspect, we make the following assumptions on the use of a production unit (M_i, B_i):

(i) Buffer B_i is used to accommodate a part (say, P_j) that has finished its processing on M_i if and only if machine M_i is scheduled to be loaded with the next part (P_{j+1}) before part P_j leaves the production unit (M_i, B_i).

(ii) The robot is not allowed to load part P_{j+1} on M_i until part P_j is moved securely to the buffer. This is done for safety – to avoid any collision between the robot and the local material handling device.

Assumption (i) implies that the use of a buffer is cycle-dependent. That is, the decision on whether a part P_j that has finished its pro-

cessing on a machine M_i is to be transferred to buffer B_i depends on the next robot operation scheduled for unit Z_i. Specifically, if the next operation is "load machine M_i," then part P_j goes to B_i immediately on the completion of its processing on M_i; thus, machine M_i is empty and ready to take the next part P_{j+1}. Otherwise, when there is no need to use a buffer, part P_j occupies machine M_i until the robot unloads it. The idea is to optimize the use of the buffer; the time to unload a part from a machine and move it to its buffer is spent only when it is necessary for the execution of the cycle. The following parameters are needed for this model:

- p_i: the processing time of a part on machine M_i, $i = 1, 2, \ldots, m$.
- ϵ: the time taken by the robot to pick up/drop off a part at I/O. Also the time taken by the robot to perform the load/unload operation at any production unit Z_i, $i = 1, 2, \ldots, m$ (i.e., load/unload machine M_i or unload buffer B_i).
- δ: the time taken by the robot to travel between two consecutive production units Z_{j-1} and Z_j, $1 \leq j \leq m + 1$. The travel times are additive for nonconsecutive machines/buffers.
- ϕ: the time taken by the robot to travel from a machine M_i (after it loads this machine) to this machine's buffer B_i.
- ω: the total time taken by the local handling device to unload a finished part from a machine M_i and transfer it to this machine's buffer B_i.

We conduct our analysis of the problem under the following assumption:

$$\omega = \phi + \epsilon. \tag{4.14}$$

Our choice of ω is based on observations in real-world robotic cells. It turns out that this assumption also slightly simplifies the analysis of the problem. Nevertheless, all the results obtained below for $RF^1_{m,\bar{1}}|(\textit{free,A,cyclic-1})|\mu$ under assumption (4.14) can be easily extended for an arbitrary value of ω. Brauner et al. [23] considered a special case of $RF^1_{m,\bar{1}}|(\textit{free,A,cyclic-1})|\mu$ in which $\phi = \omega = 0$. We also note that the model considered in this section can be easily converted to one in which

a part always goes to a machine's buffer after its processing on that machine is completed. The latter model is obtained by setting $p_i^{new} = p_i + \omega$ and $\omega^{new} = 0$.

We now discuss the construction and notational representation of 1-unit cycles for $RF^1_{m,\bar{1}}|(free,A,cyclic\text{-}1)|\mu$. We denote by M_k^- (M_k^+) and B_k^- (B_k^+), the operations "load M_k" ("unload M_k") and "load B_k" ("unload B_k"), respectively. We write $M_0^+ = I$ and $M_{m+1}^- = O$ to denote the operations "pick up a part from Input" and "drop a part at Output," respectively. Furthermore, we write Z_k^- and Z_k^+ to refer to the operations "load unit Z_k" and "unload unit Z_k," respectively. Here, Z_k refers to M_k or B_k as appropriate. We refer to the sequence of robot activities "Unload a part P_j from unit Z_k, go to machine M_{k+1}, and load part P_j on machine M_{k+1}" as *activity* A_k, and use the following notation

$$A_k = \left(Z_k^+ - \circ - M_{k+1}^-\right), \tag{4.15}$$

where $Z_k^+ \in \left\{M_k^+, B_k^+\right\}$ and $\circ \in \left\{\emptyset, \left[M_{k+1}^+, B_{k+1}^-\right]\right\}$. Here, the symbol "$\circ$" is used to specify whether or not machine M_{k+1} had to be earlier served by the local material handling device that would unload a part previously processed on M_{k+1} and move it to buffer B_{k+1}. In what follows, we will exploit the representation (4.15) in both its general form $A_k = \left(Z_k^+ - \circ - M_{k+1}^-\right)$, and a specific form where the Z_k^+ and "$\circ$" terms are explicitly specified. For example, consider the following two 1-unit cycles for problem $RF^1_{2,\bar{1}}|(free,A,cyclic\text{-}1)|\mu$.

Cycle C_1: $\left(\left(I - \left[M_1^+, B_1^-\right] - M_1^-\right), \left(B_1^+ - M_2^-\right), \left(M_2^+ - O\right)\right)$. The activities of the robot for this cycle are as follows: the robot picks a part (P_j) from Input (ϵ); goes to machine M_1 (δ); if necessary, waits for a part (P_{j-1}) on M_1 to be completed and transferred to buffer B_1 (wait time w_1); loads part P_j onto M_1 (ϵ); moves to buffer B_1 (ϕ); unloads part P_{j-1} from B_1 (ϵ); goes to machine M_2 (δ); loads part P_{j-1} onto M_2 (ϵ); waits for p_2 time units for part P_{j-1} on M_2 to be processed; unloads part P_{j-1} from M_2 (ϵ); goes to I/O (δ); and drops part P_{j-1} onto Output (ϵ). Under assumption (4.14), the cycle time of cycle C_1 is

$$T(C_1) = \max\left\{p_1 + \phi + 2\epsilon, p_2 + \phi + 3\delta + 6\epsilon\right\}.$$

Cycle C_2: $\left(\left(I-\left[M_1^+,B_1^-\right]-M_1^-\right),\left(M_2^+-O\right),\left(B_1^+-M_2^-\right)\right)$. In this cycle, the robot picks a part (P_j) from Input (ϵ); goes to machine M_1 (δ); waits, if necessary, for a part (P_{j-1}) on M_1 to be completed and transferred to buffer B_1 (w_1); loads part P_j onto M_1 (ϵ); goes to machine M_2 (δ); waits, if necessary, for a part (P_{j-2}) on M_2 to be completed (w_2); unloads part P_{j-2} from M_2 (ϵ); goes to I/O (δ); drops part P_{j-2} onto Output (ϵ), goes to buffer B_1 (δ); unloads part P_{j-1} from B_1 (ϵ); goes to machine M_2 (δ); loads part P_{j-1} onto M_2 (ϵ); and goes to I/O (δ). Under assumption (4.14), the cycle time of cycle C_2 is

$$T(C_2)=\max\left\{p_1+\phi+2\epsilon,p_2+3\delta+4\epsilon;6\delta+6\epsilon\right\}.$$

In any 1-unit cycle, each of the operations "load machine M_k," $k=1,2,\ldots,m+1$, is performed exactly once. Furthermore, for any $k=0,1,\ldots,m$, the robot operation "unload a part P_j from unit Z_k" is always immediately followed by "load part P_j onto machine M_{k+1}," i.e., after picking up part P_j from unit Z_k, the robot has no choice for its next operation but to go to machine M_{k+1} and load the latter with part P_j. We thus have the following property.

PROPERTY 4.1 *For $RF^1_{m,\bar{1}}|$(free,A,cyclic-1)$|\mu$, any feasible 1-unit cycle corresponds to a unique sequence of activities $A_k=(Z_k^+-\circ-M_{k+1}^-)$, $k=0,1,\ldots,m$, with each activity occurring exactly once (under the proviso that the latter sequence is treated in a cyclic manner).*

The next result establishes the reverse relationship between 1-unit cycles and activities.

PROPERTY 4.2 *For $RF^1_{m,\bar{1}}|$(free,A,cyclic-1)$|\mu$, any permutation of activities A_k, $k=0,1,\ldots,m$, in the form (4.15), where each $Z_k^+\in\left\{M_k^+,B_k^+\right\}$ is specified (i.e., an activity A_k is in the form of either $A_k=(M_k^+-\circ-M_{k+1}^-)$ or $A_k=(B_k^+-\circ-M_{k+1}^-)$), defines a unique 1-unit cycle.*

Proof. We first observe that by specifying $Z_k^+\in\left\{M_k^+,B_k^+\right\}$ in activity A_k, we determine how the buffer B_k is used. In particular, if in A_k we have $Z_k^+=B_k^+$, i.e., $A_k=\left(B_k^+-\circ-M_{k+1}^-\right)$, then the "$\circ$"-term of activity A_{k-1} must be $\left[M_k^+,B_k^-\right]$, i.e. $A_{k-1}=\left(Z_{k-1}^+-\left[M_k^+,B_k^-\right]-M_k^-\right)$.

Otherwise, if $Z_k^+ = M_k^+$ (so that $A_k = \left(M_k^+ - \circ - M_{k+1}^-\right)$), then $A_{k-1} = \left(Z_{k-1}^+ - M_k^-\right)$ ("$\circ$"-term is empty). Thus, by specifying the Z_k^+-entries of A_k, we uniquely identify the "$\circ$" terms of all activities and, therefore, both the robot activities and the operations within the production units are well defined. Hence, a permutation of activities A_k, $k = 0, 1, \ldots, m$, in the form (4.15), where $Z_k^+ \in \left\{M_k^+, B_k^+\right\}$, completely defines a unique cycle. We leave it to the reader to verify that such a cycle is always feasible. ∎

The following lemma gives the total number of 1-unit cycles for an m-machine single-gripper robotic cell with unit-capacity machine buffers.

LEMMA 4.3 *For $RF_{m,\bar{1}}^1|$(free,A,cyclic-1)$|\mu$, the number of 1-unit cycles equals $m! \times 2^m$.*

Proof. Clearly, any 1-unit cycle for $RF_{m,\bar{1}}^1|$(*free,A,cyclic-1*)$|\mu$, if represented by a permutation of activities A_k, $k = 0, 1, \ldots, m$, admits $(m+1)$ different representations, depending on which activity is chosen as the starting activity. To achieve the uniqueness of cycle representation, let us demand that the cycle always start with $A_0 = \left(I - \circ - M_1^-\right)$; with this assumption, by Property 4.1, a 1-unit cycle admits a unique representation. This representation is defined by a permutation of the remaining m activities A_k, $k = 1, 2, \ldots, m$, as well as by the Z_k^+-entries of activities A_k. There are $m!$ different permutations of activities A_k, $k = 1, 2, \ldots, m$, written in the form $\left(Z_k^+ - \circ - M_{k+1}^-\right)$. As Z_k^+ may stand for either M_k^+ or B_k^+, any permutation of activities $A_k = \left(Z_k^+ - \circ - M_{k+1}^-\right)$, $k = 1, 2, \ldots, m$, expands into 2^m different permutations of A_k with explicitly specified $Z_k^+ \in \left\{M_k^+, B_k^+\right\}$. By Property 4.2, any such permutation of activities A_k defines a unique 1-unit cycle for $RF_{m,\bar{1}}^1|$(*free,A,cyclic-1*)$|\mu$. ∎

Let

$$D_\phi = \{k | p_k \le \phi, \quad 1 \le k \le m\}, \quad D_\phi^c = M \setminus D_\phi, \quad r = |D_\phi^c|. \quad (4.16)$$

LEMMA 4.4 *For $RF_{m,\bar{1}}^1|$(free,A,cyclic-k)$|\mu$ with $\phi \le \delta$, a lower bound on the per unit cycle time of any k-unit cycle C is given below:*

$$\frac{T(C)}{k} \ge \max\left\{\max_{1\le i\le m} p_i + \phi + 2\epsilon; \sum_{i\in D_\phi} p_i + r\phi + (m+1)(\delta + 2\epsilon)\right\}. \quad (4.17)$$

Proof. The proof is similar to an analogous result obtained by Brauner et al. [23]. Let C be an arbitrary k-unit cycle for $RF^1_{m,\bar{1}}|(free,A,cyclic\text{-}k)|\mu$ written in terms of activities A_i, $i = 0, 1, \ldots, m$. Note that each activity $A_i = \left(Z_i^+ - \circ - M_{i+1}^-\right)$, $i = 0, 1, \ldots, m$, occurs in C exactly k times. We first estimate cycle time as the time spent by the robot to execute the cycle. To execute an activity $A_i = \left(Z_i^+ - \circ - M_{i+1}^-\right)$, the robot needs at least $2\epsilon + \delta$ time units: operations Z_i^+ and M_{i+1}^- take ϵ time each, and the robot travel time from Z_i to M_{i+1} is δ. The time τ spent by the robot between two successively executed activities, say A_i and A_l, where $A_i = \left(Z_i^+ - \circ - M_{i+1}^-\right)$ is immediately followed by $A_l = \left(Z_l^+ - \circ - M_{l+1}^-\right)$, is estimated as follows:

$$\begin{array}{ll}
\tau = 0 & \text{if } M_{i+1}^- = O, Z_l^+ = I; \\
\tau = p_{i+1} & \text{if } \quad Z_l^+ = M_{i+1}^+; \\
\tau = \phi & \text{if } \quad Z_l^+ = B_{i+1}^+; \\
\tau = \text{robot travel time} & \\
\quad \text{from } M_{i+1} \text{ to} Z_l \ \geq \delta & \text{otherwise.}
\end{array}$$

We thus have

$$T(C) \geq k \times \left(\sum_{i=1}^{m} \min\left\{p_i, \phi, \delta\right\} + (m+1)(\delta + 2\epsilon) \right).$$

As $\phi \leq \delta$, $\min\left\{p_i, \phi, \delta\right\} = \min\left\{p_i, \phi\right\}$. Hence, we obtain

$$\frac{T(C)}{k} \geq \sum_{i=1}^{m} \min\left\{p_i, \phi\right\} + (m+1)(\delta + 2\epsilon) = \sum_{i \in D_\phi} p_i + r\phi + (m+1)(\delta + 2\epsilon). \tag{4.18}$$

We now look at cycle time from a different point of view. Consider any arbitrary machine M_i, $i \in \{1, 2, \ldots, m\}$. Let τ' be the minimum time spent in the partial execution of cycle C between any two successive occurrences of M_i^- and M_i^+. Furthermore, let τ'' denote the minimum time spent in the partial cycle execution between the moment an operation M_i^+ starts and the moment the next successive operation M_i^- finishes. In the k-unit cycle C, each of the operations "load M_i" and "unload M_i" is executed k times. Hence,

$$T(C) \geq k \times \left(\tau' + \tau''\right). \tag{4.19}$$

We now estimate the values of τ' and τ''. Clearly, $\tau' \geq p_i$. As for τ'', we have two possibilities. If the corresponding cycle subsequence is $\left(\left[M_i^+ - B_i^-\right] - M_i^-\right)$, i.e., the subsequence that delivers the minimal value for τ'', then $\tau'' = \omega + \epsilon = \phi + 2\epsilon$. Otherwise, the corresponding cycle subsequence takes the form $\left(M_i^+ - \circ - M_{i+1}^- \ldots Z_{i-1}^+ - M_i^-\right)$, and τ'' must include the time to perform load/unload M_i (2ϵ) as well as the robot travel time from M_i to some other machine(s) $(\geq \delta)$ and back. Therefore, $\tau'' \geq \delta + 2\epsilon$. Thus, under the condition $\phi \leq \delta$, we always have $\tau'' \geq \phi + 2\epsilon$. By combining the above estimates for τ' and τ'' with (4.19), we deduce

$$\frac{T(C)}{k} \geq p_i + \phi + 2\epsilon, \quad i = 1, 2, \ldots, m.$$

Together with (4.18), we have the desired bound (4.17). ∎

Algorithm OPT2

Input: An instance of $RF^1_{m,\bar{1}}|(\textit{free,A,cyclic-1})|\mu$. The sets D_ϕ and D^c_ϕ are as defined by (4.16).

Output: A 1-unit cycle $\bar{C}$.

Initial setting: Units Z_k, $k \in D^c_\phi$, are occupied with a part; units Z_k, $k \in D_\phi$, are empty. The robot is positioned at I/O and the robot's gripper is empty.

Step 1: The robot picks up a part from the input buffer at the I/O hopper.

Step 2: For k from 1 to m :

Step 2.1: The robot moves to machine M_k.

Step 2.2: If $k \in D^c_\phi$, then

- if necessary, the robot waits for w_k time units for a part (P_i) on M_k to be completed and transferred to its output buffer B_k;
- the robot loads a part (P_{i+1}) onto M_k (ϵ);
- the robot moves to buffer B_k (ϕ);
- the robot unloads the part (P_i) from B_k (ϵ).

Step 2.3: Otherwise (i.e., if $k \in D_\phi$),

- the robot loads a part (P_i) onto M_k (ϵ);
- the robot waits for p_k time units for the part (P_i) on M_k to be processed;
- the robot unloads the part (P_i) from M_k (ϵ).

Step 3: The robot moves to I/O hopper and drops a part onto the output device of the I/O hopper.

THEOREM 4.6 *For $RF^1_{m,\bar{1}}|$(free,A,cyclic-k)$|\mu$ with $\phi \leq \delta$, algorithm OPT2 produces an optimal cycle $\bar{C}$. The cycle time of cycle $\bar{C}$ is*

$$T(\bar{C}) = \max\left\{\max_{k\in D^c_\phi} p_k + 2\epsilon + \phi;\ \sum_{k\in D_\phi} p_k + r\phi + (m+1)\delta + 2(m+1)\epsilon\right\}. \tag{4.20}$$

Proof. The proof of (4.20) is similar to that of Theorem 4.5. The optimality of $\bar{C}$ follows from Lemma 4.4. ∎

Observe that the role of θ in $RF^2_m|$(free,A,cyclic-1)$|\mu$ is somewhat analogous to that of ϕ in $RF^1_{m,\bar{1}}|$(free,A,cyclic-1)$|\mu$.

COROLLARY 4.3 *If $\max\{\theta,\phi\} \leq \delta$, problems $RF^2_m|$(free,A,cyclic-1)$|\mu$ and $RF^1_{m,\bar{1}}|$(free,A,cyclic-k)$|\mu$ have the same minimum cycle time if $\theta = \phi$.*

Proof. Follows from Corollary 4.2 and Theorem 4.6. ∎

COROLLARY 4.4 *Under conditions $\max\{\theta,\phi\} \leq \delta$ and $p_i \geq \delta$, $i = 1,\ldots,m$, problems $RF^2_m|$(free, A,cyclic-k)$|\mu$ and $RF^1_{m,\bar{1}}|$(free,A,cyclic-k)$|\mu$ have the same minimum cycle time if $\theta = \phi$.*

Proof. Follows from Theorems 4.5 and 4.6. ∎

We conclude this section on the relationship between problems $RF^2_m|$(free, A, cyclic-1)$|\mu$ and $RF^1_{m,\bar{1}}|$(free,A,cyclic-1)$|\mu$ with a simple result that establishes the relative equivalence of these two problems. In what follows, the robot's travel between any two machines is called an *empty move* if it travels empty, and is called a *loaded move* otherwise. Furthermore, for

$RF^1_{m,\bar{1}}|(free,A,cyclic\text{-}k)|\mu$, a production unit Z_i is said to be *fully loaded* if both M_i and B_i are loaded with a part.

DEFINITION 4.1 *A cycle C' for $RF^2_m|(free,A,cyclic\text{-}1)|\mu$ is called simple if in C' the robot never carries two parts in any of its loaded moves.*

DEFINITION 4.2 *A cycle C'' for $RF^1_{m,\bar{1}}|(free,A,cyclic\text{-}1)|\mu$ is called simple if in C'' the robot never makes an empty move from a fully loaded unit.*

Let Ω' (resp., Ω'') denote the set of all simple 1-unit cycles for $RF^2_m|(free, A, cyclic\text{-}1)|\mu$ (resp., $RF^1_{m,\bar{1}}|(free,A,cyclic\text{-}1)|\mu$). We note that a trivial example of a simple cycle in either Ω' or Ω'' is a cycle for $RF^1_m|(free, A,cyclic\text{-}1)|\mu$.

LEMMA 4.5 *Let $\theta = \phi$. Then for any cycle $C' \in \Omega'$, there exists a cycle $C'' \in \Omega''$ such that*

$$T(C') = T(C''), \tag{4.21}$$

and vice-versa.

Proof. Cycles $C' \in \Omega'$ and $C'' \in \Omega''$ that satisfy (4.21) are obtained from each other as follows. Both cycles are defined by the very same schedule of robot operations with only one exception: the sequence of operations "unload machine M_k, switch grippers, and load machine M_k" in C' translates into "load machine M_k, go to buffer B_k, and unload buffer B_k" for cycle C''. We leave it to the reader to verify that for given problem data (i.e., for given values of p_i, $i = 1, 2, \ldots, m$, δ, ϵ, and $\theta = \phi$), both cycles deliver the same cycle time. ∎

The above result shows the relative equivalence of $RF^1_{m,\bar{1}}|(free,A, cyclic\text{-}1)|\mu$ and $RF^2_m|(free,A,cyclic\text{-}1)|\mu$, when the utilization of machine buffers (for $RF^1_{m,\bar{1}}|(free,A,cyclic\text{-}1)|\mu$) and that of the additional robot gripper (for $RF^2_m|(free, A, cyclic\text{-}1)|\mu$) is limited to swapping parts at the machines. To provide intuition, we give an example of two 1-unit cycles $C' \in \Omega'$ and $C'' \in \Omega''$ that comply with Lemma 4.5:

Cycle C': $I - M_1^- - M_1^+ - M_2^- - M_3^+ - M_4^+ - M_4^- - O - M_2^+ - M_3^-$;

Cycle C'': $I - M_1^- - M_1^+ - M_2^- - M_3^+ - M_4^- - B_4^+ - O - M_2^+ - M_3^-$.

Finally, we look at the possible impact of multi-unit cycles on the cycle time for both $RF_m^2|(free,A,cyclic\text{-}k)|\mu$ and $RF_{m,\bar{1}}^1|(free,A,cyclic\text{-}k)|\mu$.

Consider the following 2-unit cycle $\hat{C}$ for $RF_m^2|(free,A,cyclic\text{-}2)|\mu$.

A 2-unit cycle $\hat{C}$: The initial state has all of the machines and the robot empty, with the robot positioned at I/O. The robot then picks up two parts (say P_i and P_{i+1}) at I/O, then moves to M_1, to M_2, ..., to M_m, and finally returns to I/O to unload both parts P_i and P_{i+1}. At each machine, P_i is loaded, processed, and unloaded, and then P_{i+1} is loaded, processed, and unloaded. The cycle time of $\hat{C}$ is

$$T(\hat{C}) = (m+1)\delta + 4(m+1)\epsilon + (m+1)\theta + 2\sum_{i=1}^{m} p_i.$$

Let x be a small positive number and X be sufficiently large such that $0 < x << X$. Consider the following problem data:

$$\theta = \phi = \epsilon = p_i = x, \quad i = 1, 2, \ldots, m; \quad \delta = X. \tag{4.22}$$

For this instance, the cycle time of $\hat{C}$ is $T(\hat{C}) = (m+1)\,X + (7m+5)\,x$. For the data in (4.22), let $OPT(P)$ denote the optimal per unit cycle for a problem P. For problem $RF_m^2|(free,A,cyclic\text{-}2)|\mu$, cycle $\hat{C}$ delivers an optimal solution for the data (4.22). That is, we have

$$OPT(RF_m^2|(free,\ A,\ cyclic\text{-}2)|\mu\ ) = \tfrac{T(\hat{C})}{2} = \tfrac{1}{2}\left((m+1)\,X + (7m+5)\,x\right).$$

Now, consider problems $RF_{m,\bar{1}}^1|(free,\ A,\ cyclic\text{-}k)|\mu$ and $RF_m^2|(free,\ A,\ cyclic\text{-}1)|\mu$. By Theorems 4.5 and 4.6, we have

$$\begin{aligned} OPT(RF_m^2|(free,A,cyclic\text{-}1)|\mu) &= OPT(RF_{m,\bar{1}}^1|(free,A,cyclic\text{-}k)|\mu) \\ &= (m+1)\,X + (3m+2)\,x. \end{aligned}$$

As $x \to 0$ and $X \to \infty$, we have

$$\begin{aligned} \frac{OPT(RF_m^2|(free,A,cyclic\text{-}1)|\mu)}{OPT(RF_m^2|(free,A,cyclic\text{-}2)|\mu)} &= \frac{OPT(RF_{m,\bar{1}}^1|(free,A,cyclic\text{-}k)|\mu)}{OPT(RF_m^2|(free,A,cyclic\text{-}2)|\mu)} \\ &= \frac{2((m+1)X + (3m+2)x)}{(m+1)X + (7m+5)x} \longrightarrow 2. \end{aligned}$$

As a consequence of the above example and Corollary 4.3, we state the following result.

LEMMA 4.6 *Under conditions $\theta \leq \delta$ and $\theta = \phi$, the long-term average throughput of a dual-gripper cell ($RF_m^2|$(free,A,cyclic-k)$|\mu$) is equal to or greater than that of the single-gripper cell with a unit-capacity output buffer at each machine ($RF_{m,\bar{1}}^1|$(free,A,cyclic-k)$|\mu$).*

To summarize, an output buffer at a machine plays the same role as an extra robot gripper – they both allow for temporary storage of a part. While the use of output buffers at the machines offers time-flexibility in part storage (as we are not directly constrained on how long a part can reside in a buffer), the use of an additional robot gripper makes storage more time-efficient (as we can save on the robot travel time). In general, an assessment of their comparative efficiency depends on the cell parameters and problem data. Finally, we note that all the results obtained in Sections 4.7 and 4.8 for additive travel-time cells will also hold for constant travel-time cells (see Chapter 2).

4.9 Dual-Gripper Robotic Cells: Constant Travel Time

In this section, we consider throughput optimization in constant-travel-time dual-gripper robotic cells (i.e., problem $RF_m^2|$(free,C,cyclic-k)$|\mu$); the results we present are from Geismar et al. [61]. We provide a structural analysis of cells with one machine per processing stage to obtain a lower bound on the throughput. Subsequently, we obtain an optimal solution under conditions that are common in practice. Unlike the previous sections of this chapter, we now consider cells where I and O are at separate locations (Figure 4.5). This is not a major change; all the results we present also hold for the layout in which I and O are at one location (with slight changes in the expressions developed for the cycle times and their lower bounds).

We start by describing a cycle $\bar{C}_{3,10}$ for two-machine cells that is similar to cycle $C_{3,10}$ of Section 4.2:

$$\bar{C}_{3,10} = (R_0^+(1,0), R_1^+(1,2), R_1^-(0,2), R_2^+(3,2), R_2^-(3,0), R_3^-(0,0))$$

At the start of the jth iteration of this cycle, the robot is at the input buffer and holds no part. Part P_{j-2} is being processed at M_2 and part P_{j-1} is being processed at M_1. The robot unloads part P_j from the input buffer and travels to M_1. If necessary, the robot waits at M_1 until

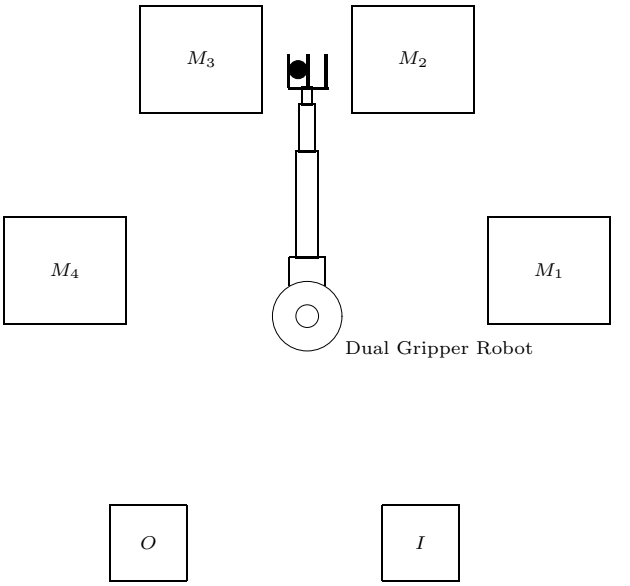


Figure 4.5. A Four-Machine Dual-Gripper Robot Cell with Separate Input and Output Buffers.

it completes processing part P_{j-1}. The robot then unloads part P_{j-1} from M_1, switches the positions of its grippers, and loads part P_j onto M_1. Next, the robot carries part P_{j-1} to M_2. At M_2, the robot waits until it completes processing part P_{j-2}, if necessary. The robot then unloads part P_{j-2} from M_2, switches the positions of its grippers, and loads part P_{j-1} onto M_2. The robot then travels to the output buffer and loads part P_{j-2} onto it. The robot completes the cycle by returning to I. It is straightforward to verify that the cycle time of $\bar{C}_{3,10}$ equals $4\delta + 6\epsilon + 2\theta + w_1 + w_2$, where w_1 and w_2 are the robot waiting times that can be determined from the steady-state conditions.

Instead of analyzing $\bar{C}_{3,10}$, we study its m-machine generalization $\bar{C}_d^m$. As expected, $\bar{C}_d^m$ is similar to cycle C_d^m of Section 4.3; the robot moves are slightly different at I and O. Cycle $\bar{C}_d^m$ starts with the state in which all machines are occupied with parts and the robot is empty at I. The sequence of activities for the robot in this cycle is as follows:

Cycle $\bar{C}_d^m$
Begin
    ϵ: Robot unloads a part from I.
    For $i = 1$ to m do:
    Begin
        δ: Robot moves to M_i.
        w_i: Robot waits for the part on M_i to be completed.
        ϵ: Robot unloads M_i.
        θ: Robot switches to the other gripper.
        ϵ: Robot loads M_i.
    End (Next i)
    δ: Robot moves to O.
    ϵ: Robot unloads finished part at O.
    δ: Robot moves to I.
End

The cycle time for $\bar{C}_d^m$ can be easily calculated as

$$T(\bar{C}_d^m) = (m+2)\delta + 2(m+1)\epsilon + m\theta + \sum_{j=1}^{m} w_j. \tag{4.23}$$

The total waiting time $\sum_{j=1}^{m} w_j$ can be derived as in Section 4.3. Substituting, we obtain the following expression for the cycle time:

$$T(\bar{C}_d^m) = \max\{(m+2)\delta + 2(m+1)\epsilon + m\theta, \max_{1\le j\le m}\{p_j\} + 2\epsilon + \theta\}. \tag{4.24}$$

4.9.1 Lower Bounds and Optimal Cycles: *m*-Machine Simple Robotic Cells

Assuming $\theta \le \min\{\delta, p_1, \ldots, p_m\}$, we show that $\bar{C}_d^m$ is optimal among all dual-gripper 1-unit cyclic schedules (Corollary 4.5). We then show that $\bar{C}_d^m$ is optimal among all k-unit cycles ($k \ge 1$) under an additional restriction that is quite common in practice: $p_i \ge \delta$, $\forall i$ (Corollary 4.6). Intuitively, $\bar{C}_d^m$ is optimal because it minimizes inter-machine travel (each machine is visited once per cycle), and minimizes the time that the robot spends at each machine waiting for it to complete processing: after loading a particular machine M_i, the robot visits every other machine before returning to M_i to unload it.

4.9.2 One-Unit Cycles

We study here the lower bounds for problem $RF_m^2|(free,C,cyclic\text{-}1)|\mu$. To show that $\bar{C}_d^m$ is an optimal 1-unit cycle under $\theta \leq \min\{\delta, p_1, \ldots, p_m\}$, we establish a lower bound that is equal to $T(\bar{C}_d^m)$ on such cycles. A similar lower bound for additive travel-time cells was established in Section 4.4.

THEOREM 4.7 *Assume $\theta \leq \min\{\delta, p_1, \ldots, p_m\}$. A lower bound for cycle times for 1-unit robot move cycles in a constant travel-time dual-gripper cell is given by*

$$LB = \max\{(m+2)\delta + 2(m+1)\epsilon + m\theta, \max_{1\leq i\leq m}\{p_i\} + 2\epsilon + \theta\}. \quad (4.25)$$

Proof. In part A of the proof, we show that $\max\{p_i\} + 2\epsilon + \theta$ is a lower bound. In part B, we show that $(m+2)\delta + 2(m+1)\epsilon + m\theta$ is a lower bound.

Part A: Consider any machine M_i $(1 \leq i \leq m)$ and a 1-unit cyclic schedule π. Note that π can be represented by some feasible ordering of the symbols $\mathcal{M} = \{M_0^+, M_1^+, \ldots, M_m^+, M_1^-, M_2^-, \ldots, M_{m+1}^-\}$. Since π is cyclic, we may assume it to be of the form $(M_i^-, \sigma_1, M_i^+, \sigma_2)$ for any i, where σ_1 and σ_2 are feasible subschedules and $\widehat{\sigma}_h \subset \mathcal{M}$ is the set of activities in σ_h, $h = 1, 2$, such that $\widehat{\sigma}_1 \cap \widehat{\sigma}_2 = \emptyset$ and $\widehat{\sigma}_1 \cup \widehat{\sigma}_2 = \mathcal{M} \setminus \{M_i^-, M_i^+\}$. Two cases are considered.

Case 1. $\widehat{\sigma}_2 \neq \emptyset$. The time from the start of loading of a part onto M_i until the completion of unloading from M_i is at least $p_i + 2\epsilon$ (covering the subschedule (M_i^-, σ_1, M_i^+)). The robot will be engaged for at least an additional amount of time 2δ in order to complete σ_2. Thus, a lower bound on the schedule length is $p_i + 2\epsilon + 2\delta \geq p_i + 2\epsilon + \theta$.

Case 2. $\widehat{\sigma}_1 \neq \emptyset, \widehat{\sigma}_2 = \emptyset$. This case can occur only with a dual-gripper robot. The cycle may be denoted by (M_i^+, M_i^-, σ_1). The subcycle (M_i^+, M_i^-) requires $2\epsilon + \theta$ time units and the delay from the completion of M_i^- to the start of M_i^+ is of length at least p_i, for a total time of $p_i + 2\epsilon + \theta$ time units.

Cases 1 and 2 prove that $\max\{p_i\} + 2\epsilon + \theta$ is a lower bound on the cycle times.

Part B: By using that the cycle π is of the form $(M_i^-, \sigma_1, M_i^+, \sigma_2)$, where at least one of $\widehat{\sigma}_1$ and $\widehat{\sigma}_2$ is not empty, we shall compute lower bounds for the aggregate residence times of the robot at each machine and for the aggregate robot transportation times between the machines. By *residence times* we mean the times during which the robot is occupied at a machine while either waiting for that machine to complete processing, rotating its grippers, or loading or unloading the machine. The sum of these computed lower bounds will give us the bound of $(m+2)\delta + 2(m+1)\epsilon + m\theta$. In our discussion, $p_0 = p_{m+1} = 0$ is the "processing time" at I and O. For residence times at the machines, there are five cases to consider.

Case 0. $M_i = M_0$ (the "machine" is I). Since we do not have loading on M_0, the sequence is simply (M_0^+, σ_2). The robot is occupied at M_0 for ϵ time units.

Case 1. $M_i = M_{m+1}$ (the "machine" is O). Since we do not have unloading on M_{m+1}, the sequence is simply (M_{m+1}^-, σ_1). The robot is occupied at M_{m+1} for ϵ time units.

Case 2. $\widehat{\sigma}_2 = \emptyset$. The robot is occupied at M_i for at least $2\epsilon + \theta$ time units.

Case 3. $\widehat{\sigma}_1 = \emptyset$. The robot is occupied at M_i for at least $2\epsilon + p_i$ time units.

Case 4. $\widehat{\sigma}_1 \neq \emptyset \neq \widehat{\sigma}_2$. The robot is occupied at M_i for at least 2ϵ time units (split between two visits).

Let u_j be the number of machines included in Case j and let U_j be the set of machines included in Case j, $j = 0, 1, 2, 3, 4$. Note that $u_2 + u_3 + u_4 = m$ and $u_0 + u_1 = 2$. We now have the following lower bound on the aggregate residence time for the robot at all machines:

$$(u_0+u_1)\epsilon + u_2(2\epsilon+\theta) + u_3(2\epsilon) + \sum_{i\in U_3} p_i + u_4(2\epsilon) = 2(m+1)\epsilon + u_2\theta + \sum_{i\in U_3} p_i. \tag{4.26}$$

A robot movement from M_i to M_h occurs when an operation M_i^- or M_i^+ is followed immediately by a robot operation M_h^- or M_h^+, where $h \neq i$. Such a movement is *incident* to both M_i and M_h, and requires δ time units.

For each machine M_i in Case 0, 1, 2, or 3, there is at least one movement to and one movement away from M_i, so M_i is incident to at least two movements. If M_i is in Case 4, then each cycle includes at least two movements to and two movements away from M_i, so M_i will be incident to at least four movements. Thus, the aggregate of incidences over all machines $M_0, M_1, \ldots, M_{m+1}$ is at least $2(u_0 + u_1 + u_2 + u_3) + 4u_4 = 2(m+2) + 2u_4$. Hence, the total number of movements, which is half the aggregate incidences, is at least $m+2+u_4$. These movements require a minimum aggregate time of $(m+2+u_4)\delta$. A lower bound on the cycle times is obtained by adding this to (4.26). The resulting expression, since $\theta \leq \min\{\delta, p_1, \ldots, p_m\}$, can be simplified to give a lower bound of $(m+2)\delta + 2(m+1)\epsilon + m\theta$. ∎

COROLLARY 4.5 *In a simple robotic cell with a dual-gripper robot, $\bar{C}_d^m$ is optimal among all 1-unit cyclic schedules under the assumption that $\theta \leq \min\{\delta, p_1, \ldots, p_m\}$.*

4.9.3 Multi-Unit Cycles

We now discuss lower bounds for problem $RF_m^2|(free,C,cyclic\text{-}k)|\mu$. In Chapter 3, we discussed examples of single-gripper cells where the throughput of an optimal 2-unit cycle is better than that of an optimal 1-unit cycle. Such examples exist for dual-gripper robotic cells too. All 1-unit cycles have the form $(M_i^-, \sigma_1, M_i^+, \sigma_2)$, where $\widehat{\sigma}_1 \cup \widehat{\sigma}_2 = \mathcal{M} \setminus \{M_i^-, M_i^+\}$ and $\widehat{\sigma}_1 \cap \widehat{\sigma}_2 = \emptyset$. So, they cannot exploit fully the capabilities of a dual-gripper robot – a dual-gripper robot can (i) unload two parts from I while resident at I, (ii) load two parts onto O while resident at O, or (iii) load M_i, wait for its processing, unload M_i, and then load M_i again, $i = 1, \ldots, m$. We consider a 2-unit cycle that takes advantage of these features.

Consider the 2-unit cycle $\hat{C}$ of Section 4.8: the initial state has all of the machines and the robot empty and the robot positioned at I. The robot then unloads two parts (say P_j and P_{j+1}) from I, then moves to M_1, to M_2, ..., to M_m, then moves to O where it places both parts

P_j and P_{j+1}, and then finally returns to I. At each machine, P_j is loaded, processed, and unloaded, and then P_{j+1} is loaded, processed, and unloaded.

The cycle time of $\hat{C}$ is $\bar{T}(\hat{C}) = (m+2)\delta + 4(m+1)\epsilon + (m+2)\theta + 2\sum_{i=1}^{m} p_i$. Since two parts are produced in this cycle, the average time to produce one part is

$$\frac{\bar{T}(\hat{C})}{2} = (m+2)\frac{\delta}{2} + 2(m+1)\epsilon + (m+2)\frac{\theta}{2} + \sum_{i=1}^{m} p_i.$$

Note that the cycle time expression above is slightly different than one obtained for $\hat{C}$ in Section 4.8, as a result of the separation of the locations for I and O in the layout considered in this section.

Observe that $\bar{T}(\hat{C})/2 < T(\bar{C}_d^m) = \max\{(m+2)\delta + 2(m+1)\epsilon + m\theta, \max\{p_j\} + 2\epsilon + \theta\}$, for $\theta = 0.5$, $\epsilon = 1$, $\delta = 4$, $p_1 = p_2 = \cdots = p_m = 1$, and that the condition $\theta \leq \min\{\delta, p_1, \ldots, p_m\}$ is also satisfied. For this data, the 2-unit cycle $\hat{C}$ is better than the best 1-unit cycle. However, $\bar{T}(\hat{C})/2 \geq T(\bar{C}_d^m)$ if $p_i \geq \delta, \forall i$, and $\theta \leq \min\{\delta, p_1, \ldots, p_m\}$.

We now show that the 1-unit cycle $\bar{C}_d^m$ is optimal among all k-unit ($k \geq 1$) cycles under two conditions (i) $\theta \leq \delta$ and (ii) $p_i \geq \delta$, $i = 1, \ldots, m$, that are common in practice (Kumar et al. [102] and Perkinson et al. [128]). Again our approach is to show that $T(\bar{C}_d^m)$ is a lower bound on the per unit cycle time in such a cell.

THEOREM 4.8 *If $\theta \leq \delta$ and $p_i \geq \delta$, $i = 1, \ldots, m$, then for any k-unit cycle π ($k \geq 1$) in a simple robotic cell served by a dual-gripper robot, the cycle time $T(\pi)$ satisfies*

$$\frac{T(\pi)}{k} \geq \max\left\{(m+2)\delta + 2(m+1)\epsilon + m\theta, \max_{1\leq i\leq m}\{p_i\} + 2\epsilon + \theta\right\}.$$

Proof. The proof that $\max\{p_i\} + 2\epsilon + \theta$ is a lower bound for $T(\pi)/k$ is similar to that for Theorem 4.7. In part B, we show that $(m+2)\delta + 2(m+1)\epsilon + m\theta$ is a lower bound for $T(\pi)/k$.

Part B: This bound is true for $k = 1$ by Theorem 4.7. Now we assume $k \geq 2$. First we prove the result for k even. Note that for any machine M_i, any sequence σ of a k-unit cycle that represents two consecutive loadings and unloadings of machine M_i is of the form $\sigma = (M_{i,2r-1}^-, \sigma_1, M_{i,2r-1}^+, \sigma_2, M_{i,2r}^-, \sigma_3, M_{i,2r}^+, \sigma_4)$, where $r = 1, \ldots, \frac{k}{2}$. $M_{i,2r}^-$

($M^+_{i,2r}$) denotes $2r^{th}$ loading (unloading) of machine M_i. Note that in σ at least one of $\widehat{\sigma}_1$, $\widehat{\sigma}_2$, $\widehat{\sigma}_3$, and $\widehat{\sigma}_4$ is not empty.

Now we establish a lower bound on the aggregate residence time of the robot at all machines in a k-unit cycle by considering 2-unit subcycles. We need to consider the following cases. Cases 0 and 1 deal with sequences $\sigma = (M^-_{i,2r-1}, \sigma_1, M^+_{i,2r-1}, \sigma_2,\ M^-_{i,2r}, \sigma_3, M^+_{i,2r}, \sigma_4)$, for $i = 0$ and $i = m+1$, respectively, whereas Cases 2-10 deal with all other i, $1 \leq i \leq m$.

Case 0. $M_i = M_0$ (the "machine" is I). Since we do not have loading onto M_0, the sequence is simply $(M^+_{0,2r-1}, \sigma_1, M^+_{0,2r}, \sigma_2)$, where at least one of $\widehat{\sigma}_1$ and $\widehat{\sigma}_2$ is not empty. If $\widehat{\sigma}_1 \neq \emptyset$ and $\widehat{\sigma}_2 \neq \emptyset$, then the robot is occupied at M_0 for 2ϵ time units. If $\widehat{\sigma}_1 = \emptyset$ (or $\widehat{\sigma}_2 = \emptyset$), then the robot is occupied at M_0 for $2\epsilon + \theta$ time units.

Case 1. $M_i = M_{m+1}$ (the "machine" is O). Since we do not have unloading from M_{m+1}, the sequence is simply $(M^-_{m+1,2r-1}, \sigma_1, M^-_{m+1,2r}, \sigma_2)$, where at least one of $\widehat{\sigma}_1$ and $\widehat{\sigma}_2$ is not empty. If $\widehat{\sigma}_1 \neq \emptyset$ and $\widehat{\sigma}_2 \neq \emptyset$, then the robot is occupied at M_{m+1} for 2ϵ time units. If $\widehat{\sigma}_1 = \emptyset$ (or $\widehat{\sigma}_2 = \emptyset$), then the robot is occupied at M_{m+1} for $2\epsilon + \theta$ time units.

Case 2. $\widehat{\sigma}_1 \neq \emptyset, \widehat{\sigma}_2 \neq \emptyset, \widehat{\sigma}_3 \neq \emptyset, \widehat{\sigma}_4 \neq \emptyset$. The robot is occupied at M_i for at least 4ϵ time units.

Case 3. $\widehat{\sigma}_1 = \emptyset$ (or $\widehat{\sigma}_3 = \emptyset$). First consider $\widehat{\sigma}_1 = \emptyset$. The robot is occupied at M_i for at least $4\epsilon + p_i$ time units. As $\widehat{\sigma}_3 = \emptyset$ has the same schedule structure as $\widehat{\sigma}_1 = \emptyset$, it will have the same machine residence time.

Case 4. $\widehat{\sigma}_2 = \emptyset$ (or $\widehat{\sigma}_4 = \emptyset$). The robot is occupied at M_i for at least $4\epsilon + \theta$ time units.

Case 5. $\widehat{\sigma}_1 = \widehat{\sigma}_2 = \emptyset$ (or $\widehat{\sigma}_3 = \widehat{\sigma}_4 = \emptyset$). The robot is occupied at M_i for at least $4\epsilon + \theta + p_i$ time units.

Case 6. $\widehat{\sigma}_1 = \widehat{\sigma}_3 = \emptyset$. The robot is occupied at M_i for at least $4\epsilon + 2p_i$ time units.

Case 7. $\widehat{\sigma}_1 = \widehat{\sigma}_4 = \emptyset$ (or $\widehat{\sigma}_2 = \widehat{\sigma}_3 = \emptyset$). The robot is occupied at M_i for at least $4\epsilon + \theta + p_i$ time units.

Case 8. $\widehat{\sigma}_2 = \widehat{\sigma}_4 = \emptyset$. The robot is occupied at M_i for at least $4\epsilon + 2\theta$ time units.

Case 9. $\widehat{\sigma}_1 = \widehat{\sigma}_2 = \widehat{\sigma}_3 = \emptyset$ (or $\widehat{\sigma}_1 = \widehat{\sigma}_3 = \widehat{\sigma}_4 = \emptyset$) The robot is occupied at M_i for at least $4\epsilon + \theta + 2p_i$ time units.

Case 10. $\widehat{\sigma}_1 = \widehat{\sigma}_2 = \widehat{\sigma}_4 = \emptyset$ (or $\widehat{\sigma}_2 = \widehat{\sigma}_3 = \widehat{\sigma}_4 = \emptyset$). This schedule structure is infeasible.

Let u_j denote the number of sequences σ corresponding to Case j that occur in a k-unit cycle, for all j and r ($j = 0, 1, \ldots, 9$, and $r = 1, \ldots, k/2$) and machines M_i, $i = 0, \ldots, m+1$. As there are $k/2$ sequences for each machine in a k-unit cycle, we have $mk/2 = u_2 + u_3 + u_4 + u_5 + u_6 + u_7 + u_8 + u_9$ and $u_0 = u_1 = k/2$. By adding residence times corresponding to all the above cases and setting $p_i = \delta$, we get a lower bound for T_r, the aggregate residence time of the robot at all machines:

$$\begin{aligned} T_r \;\geq\; & \chi + 2mk\epsilon + (u_4 + u_5 + u_7 + 2u_8 + u_9)\theta \\ & + (u_3 + u_5 + 2u_6 + u_7 + 2u_9)\delta, \end{aligned}$$

where χ denotes the total minimum cumulative residence time of the robot at both I and O in all sequences $(M^+_{0,2r-1}, \sigma_1,\ M^+_{0,2r}, \sigma_2)$ and $(M^-_{m+1,2r-1},\ \sigma_1,\ M^-_{m+1,2r}, \sigma_2)$ in the k-unit cycle.

If a machine M_i is included in Case 2, then there are in each cycle at least four movements to and four movements away from M_i, so M_i will be incident to at least eight movements. If a machine M_i is included in Case 3 or 4, there are in each cycle at least three movements to and three movements away from M_i, so M_i will be incident to at least six movements. If a machine M_i is included in Case 5, 6, 7, or 8, there are in each cycle at least two movements to and two movements away from M_i, so M_i will be incident to at least four movements. If a machine M_i is included in Case 9, there is in each cycle at least one movement to and one movement away from M_i, so M_i will be incident to at least two movements.

Thus, the aggregate of incidences over all machines is at least $2\lambda + 8u_2 + 6(u_3 + u_4) + 4(u_5 + u_6 + u_7 + u_8) + 2u_9$, where 2λ is the total number of incidences over I and O. Hence, the total number of movements, which is half the aggregate incidences, is at least $\lambda + 4u_2 + 3(u_3 + u_4) + 2(u_5 +$

$u_6+u_7+u_8)+u_9$. Since $mk/2 = u_2+\cdots+u_9$, these movements require a minimum aggregate time of

$$T_t = \lambda\delta + mk\delta + 2u_2\delta + (u_3+u_4)\delta - u_9\delta.$$

Hence,

$$\begin{aligned} T_r + T_t &\geq \chi + 2mk\epsilon + (u_4+u_5+u_7+2u_8+u_9)\theta + (u_3+u_5+2u_6 \\ &\quad +u_7+2u_9)\delta + \lambda\delta + mk\delta + +2u_2\delta + (u_3+u_4)\delta - u_9\delta \\ &\geq (\lambda\delta+\chi) + mk\delta + 2mk\epsilon + (u_4+u_5+u_7+2u_8+u_9)\theta \\ &\quad +(2u_2+2u_3+u_4+u_5+2u_6+u_7+u_9)\delta. \end{aligned}$$

We now obtain an estimate for $(\lambda\delta+\chi)$. We saw in Case 0 (Case 1) that if the robot unloads (loads) two parts in two separate visits to I (O), its total residence time is 2ϵ. In this scenario, its travel time covers four incidences, i.e., it adds 2δ to the minimum travel time. If the robot unloads (loads) two parts in one trip, its residence time is $2\epsilon+\theta$, and the added travel time is δ. Therefore,

$$\lambda\delta+\chi = u_0^{'}(2\delta+2\epsilon) + u_0^{''}(\delta+2\epsilon+\theta) + u_1^{'}(2\delta+2\epsilon) + u_1^{''}(\delta+2\epsilon+\theta),$$

where $u_0^{'}$ ($u_1^{'}$) denotes the number of Case 0 (Case 1) subsequences in which the robot unloads (loads) two parts in two separate visits to I (O), and $u_0^{''}$ ($u_1^{''}$) denotes the number of Case 0 (Case 1) subsequences in which the robot unloads (loads) two parts in one visit to I (O). Note that $u_0^{'}+u_0^{''} = k/2$ and $u_1^{'}+u_1^{''} = k/2$. Therefore,

$$\begin{aligned} \lambda\delta+\chi &= u_0^{'}(2\delta+2\epsilon) + u_0^{''}(\delta+2\epsilon+\theta) + u_1^{'}(2\delta+2\epsilon) + u_1^{''}(\delta+2\epsilon+\theta) \\ &= k\delta + 2k\epsilon + (u_0^{'}+u_1^{'})\delta + (u_0^{''}+u_1^{''})\theta. \end{aligned}$$

Thus, we have

$$\begin{aligned} T_r + T_t &\geq k\delta + 2k\epsilon + (u_0^{'}+u_1^{'})\delta + (u_0^{''}+u_1^{''})\theta + mk\delta + 2mk\epsilon \\ &\quad +(u_4+u_5+u_7+2u_8+u_9)\theta \\ &\quad +(2u_2+2u_3+u_4+u_5+2u_6+u_7+u_9)\delta \\ &= (m+2)k\delta + 2(m+1)k\epsilon + (u_4+u_5+u_7+2u_8+u_9+u_0^{''} \\ &\quad +u_1^{''})\theta + (2u_2+2u_3+u_4+u_5+2u_6+u_7+u_9-u_0^{''}-u_1^{''})\delta \\ &= (m+2)k\delta + 2(m+1)k\epsilon + mk\theta + (2u_2+2u_3+u_4+u_5 \\ &\quad +2u_6+u_7+u_9-u_0^{''}-u_1^{''})(\delta-\theta). \end{aligned} \tag{4.27}$$

Now observe that for each subsequence $(M^+_{0,2r-1}, \sigma_1, M^+_{0,2r}, \sigma_2)$, if $\widehat{\sigma}_1 = \emptyset$ or $\widehat{\sigma}_2 = \emptyset$, i.e., if the robot loads two parts in one visit to I, then the choices for its next moves are limited. Obviously, it must first load machine M_1. Furthermore, since $\delta \geq \theta$, it is only advantageous to load both grippers in one trip to I if the next sequence of moves is $M_1^- M_1^+ M_1^-$, i.e., for each time the robot has a sequence $M_0^+ M_0^+$, machine M_1 belongs either to Case 5 or to Case 9. Similarly, for each time there is a sequence $M^-_{m+1} M^-_{m+1}$, machine M_m belongs to Case 7. Hence, $u_5 + u_9 \geq u_0^{''}$ and $u_7 \geq u_1^{''}$. Therefore, the coefficient of $(\delta - \theta)$ in (4.27) is positive, so

$$T_r + T_t \quad \geq \quad (m+2)k\delta + 2(m+1)k\epsilon + mk\theta,$$

and

$$\frac{(T_r + T_t)}{k} \geq (m+2)\delta + 2(m+1)\epsilon + m\theta.$$

Since $T(\pi)/k \geq (T_r + T_t)/k$, we have the desired lower bound for k even.

We now prove the result for k odd. We have just shown that the lower bound for the first $(k-1)/2$ sequences $(M^-_{i,2r-1}, \sigma_1, M^+_{i,2r-1}, \sigma_2, M^-_{i,2r}, \sigma_3, M^+_{i,2r}, \sigma_4)$, for $r = 1, \ldots, (k-1)/2$, is

$$T_r + T_t \geq (m+2)(k-1)\delta + 2(m+1)(k-1)\epsilon + m(k-1)\theta.$$

As a consequence of Theorem 4.7, for the last sequence $\sigma^{'} = (M^-_{i,k}, \sigma_1, M^+_{i,k}, \sigma_2)$, we have the following lower bound for the robot residence times and robot move times:

$$T'_r + T'_t \geq (m+2)\delta + 2(m+1)\epsilon + m\theta.$$

By adding these two inequalities, we obtain

$$T_r + T_t + T'_r + T'_t \geq (m+2)k\delta + 2(m+1)k\epsilon + mk\theta.$$

Thus, we have the desired bound

$$\frac{T(\pi)}{k} \geq \frac{(T_r + T_t + T'_r + T'_t)}{k} \geq (m+2)\delta + 2(m+1)\epsilon + m\theta.$$

■

COROLLARY 4.6 *In a simple robotic cell with a dual-gripper robot, $\bar{C}_d^m$ is optimal among all k-unit cyclic schedules ($k \geq 1$) under the assumption that $\theta \leq \delta$ and $p_i \geq \delta$, $i = 1, \ldots, m$.*

REMARK 4.1 All the results obtained above for constant travel-time cells also hold for cells with circular layout and regular additive travel-time metric (see Chapter 2). In a general additive travel-time cell, however, the machines are typically arranged along a line (Lei and Wang [108]), or along a semicircular arc (Brauner and Finke [22]) so that the total travel time from M_{m+1} to M_0 is $(m+1)\delta$. In such a layout, the cycle time for $\bar{C}_d^m$ increases significantly; consequently, the analysis required to determine a lower bound on the per unit cycle time is fundamentally different.

REMARK 4.2 Because the lower bound for the per unit cycle time in a simple robotic cell with a single-gripper robot is (Dawande et al. [47], Chapter 3)

$$\frac{T(\pi)}{k} \geq \max\left\{2(m+1)\epsilon + \sum_{i=1}^{m}\min\{p_i,\delta\} + (m+2)\delta, \max_{1\leq i\leq m} p_i + 3\delta + 4\epsilon\right\},$$

$\bar{C}_d^m$ achieves a greater throughput than all single-gripper k-unit cycles in cells that satisfy $\theta \leq \min\{\delta, p_i\}$, $i = 1, \ldots, m$. This is not surprising, given that in cycle $\bar{C}_d^m$ the dual-gripper robot has a delay of only θ between unloading and loading a machine. A single-gripper robot, after unloading a machine M_i, must do at least the following before reloading M_i: travel to M_{i+1} (δ), load M_{i+1} (ϵ), travel to M_{i-1} (δ), unload M_{i-1} (ϵ), and travel to M_i (δ).

Chapter 5

PARALLEL MACHINES

In the classical parallel machine part-scheduling problem, jobs are processed by identical machines in parallel. Each job requires only a single operation, and it may be processed on any of those machines [132]. Hall et al. [79] analyze such systems in which all jobs must be loaded (set up) by a common server. They provide either polynomial or pseudo-polynomial algorithms, or a proof of NP-hardness for various conditions on setup times, processing times, and objectives. Błażewicz et al. [14] analyze the Vehicle Routing with Time Windows problem, in addition to the part-scheduling problem, for a similar system of parallel machines, each of which can perform various tasks. These machines are served by several automated guided vehicles that travel the same circuit.

This chapter considers constant travel-time robotic cells with parallel machines producing a single part-type. Just as a simple robotic cell is analogous to a flow shop with blocking, a robotic cell with parallel machines is analogous to a flexible flow shop with blocking. In a robotic cell with parallel machines, there are m stages, and for each processing stage i there are $m_i \geq 1$ identical machines. As with simple cells, each part is processed at each stage according to the same fixed sequence. A part can be processed at stage i by any one of the m_i machines at that stage.

The m_i distinct machines at stage i are denoted $M_{ia}, M_{ib}, \ldots, M_{i,\alpha(m_i)}$, where the function $\alpha(j)$ assigns to any positive integer j the j^{th} letter of some alphabet. For ease of exposition, we use the standard English

alphabet, e.g., $\alpha(3) = c$; if a stage i has $m_i \geq 27$ distinct machines, then any other appropriate alphabet can be used. Each machine at stage i has processing time p_i. All cells discussed in this chapter have constant travel time: $d(M_{i\gamma}, M_{j\eta}) = \delta$, if $i \neq j$ or $\gamma \neq \eta$, whether the robot is carrying a part or not. In addition, in this chapter we consider only cells that produce identical parts. Section 5.1 analyzes robotic cells with parallel machines that have single-gripper robots; cells with dual-gripper robots are considered in Section 5.2.

5.1 Single-Gripper Robots

In certain cells, throughput can be improved by adding an identical machine to a particular processing stage. Such a machine would be used in parallel with the other machines of that stage. This method is especially cost effective if there are a small number of stages whose processing times are significantly larger than those of the other stages. In fact, using m_j parallel machines at stage j reduces that stage's impact on the per unit cycle time's lower bound by a factor of m_j: $T_c(\pi)/k \geq (p_j + 3\delta + 4\epsilon)/m_j$, where cycle π produces k parts. Herrmann et al. [83] devise a network model that can be used to perform sensitivity analysis to determine the amount of reduction in the cycle time by the addition of a parallel machine to a specific stage (see Chapter 3). Our focus is on finding an optimal cycle of robot moves in a robotic cell with parallel machines (problem $RF_m(m_1, m_2, \ldots, m_m)|(free,C,cyclic\text{-}k)|\mu$).

5.1.1 Definitions

For clarity and flexibility, we define the concept of *activity* for a system of parallel machines. When transferring a part from one machine to another, the activity is denoted with three subscripts, e.g., $A_{i\gamma\eta}$. This indicates that a part is being transferred from stage i to stage $i + 1$, is being unloaded from machine $M_{i\gamma}$, and is being loaded onto machine $M_{i+1,\eta}$. If the source is I or the destination is O, instead of a letter, we use the asterisk symbol (*). For example, activity A_{1ba} means that the robot takes the part from M_{1b}, travels to M_{2a}, and loads the part onto M_{2a}. To signify taking a part from I, moving to M_{1a}, and loading M_{1a}, we write A_{0*a}.

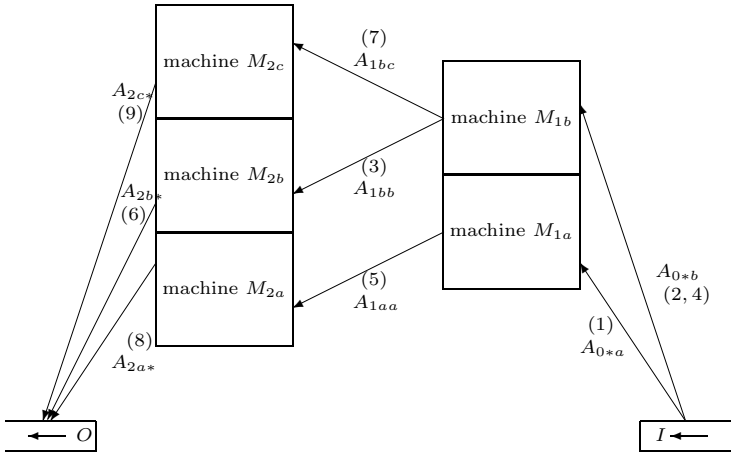


Figure 5.1. Cycle π_1 of Example 5.1 in a Two-Stage Robotic Cell with Parallel Machines. $m_1 = 2, m_2 = 3$. Numbers in Parentheses Indicate Order of Operations.

Note that the definition of a k-unit cycle (Chapter 3) can be easily adapted to a cell with parallel machines:

DEFINITION 5.1 A *k-unit cycle* π in a robotic cell with parallel machines is a feasible sequence of robot moves in which each stage has its machines loaded and unloaded exactly k times, and the cell returns to its initial state.

Hence, our definitions of cycle time and per unit cycle time are still valid.

EXAMPLE 5.1 Consider a cell with two stages ($m = 2$) with $m_1 = 2$ and $m_2 = 3$. Here is an example of a cycle that produces three ($k = 3$) parts:

$$\pi_1 = (A_{0*a}, A_{0*b}, A_{1bb}, A_{0*b}, A_{1aa}, A_{2b*}, A_{1bc}, A_{2a*}, A_{2c*}).$$

Note that this cycle has $k(m + 1) = 9$ activities. A schematic picture of this cycle can be found in Figure 5.1. A Gantt chart can be found in Figure 5.2. This cycle is feasible because the activities that load M_{1b} (A_{0*b} twice) alternate with those that unload M_{1b} (A_{1bb} and A_{1bc}).

A cycle in a cell with parallel machines can be checked for feasibility as follows. For each machine $M_{i\ell}, i \in M,\ \ell = a, \ldots, \alpha(m_i)$, between any two activities that load $M_{i\ell}$ ($A_{i-1,x\ell}$, $x \in \{a, \ldots, \alpha(m_{i-1})\}$), there must be exactly one activity that unloads $M_{i\ell}$ ($A_{i\ell y}$, $y \in \{a, \ldots, \alpha(m_{i+1})\}$).

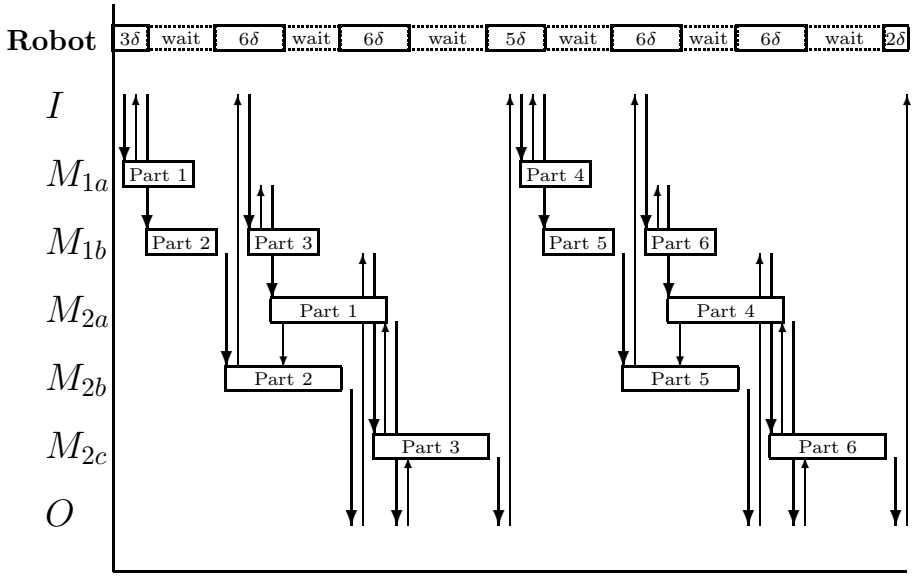


Figure 5.2. Gantt Chart for Cycle π_1 of Example 5.1. Thin Lines Indicate that the Robot Is Traveling Without a Part. In this Cell, $p_1 = 30$, $p_2 = 50$, $\delta = 5$, and $\epsilon = 0$.

5.1.2 *k*-Unit Cycles and Blocked Cycles

We now examine k-unit cycles ($k \geq 1$) for a robotic cell with parallel machines and m stages. We first derive elementary results for them. Then, we will study a special class – blocked cycles – of k-unit cycles ($k \geq 2$), and find a dominating subclass.

5.1.2.1 Structural Results for *k*-Unit Cycles

Given a cell of m stages, we establish a lower bound for the cycle time of all k-unit cycles on this cell. This lower bound is a generalization of Theorem 3.19.

THEOREM 5.1 *For any k-unit cycle π, the per unit cycle time $T(\pi)/k$ satisfies*

$$\frac{T(\pi)}{k} \geq \max\left\{2(m+1)\epsilon + \sum_{i=1}^{m} \min\{p_i, \delta\} + (m+2)\delta, \max_{1\leq i\leq m} \frac{p_i + 3\delta + 4\epsilon}{m_i}\right\}.$$

Proof. A k-unit cycle consists of $k(m+1)$ activities. Each activity requires one loading and one unloading, so the total time for these actions is $2k(m+1)\epsilon$. Before each activity A_i, $i \in M$, there will be time taken either by a robot move (δ) or a processing time (p_i). This time is represented by the second term. The robot never has to wait for processing to complete before executing an A_0, so the total time taken before all k A_0's is $k\delta$; this is included in the last term. The last term also includes the robot travel time while performing the $k(m+1)$ activities (transfer of a part from some machine in stage i to a machine in stage $i+1$, $i = 0, \ldots, m$), which is $k(m+1)\delta$. A proof of the second term is trivial; we leave it to the reader. ∎

This leads to an immediate result for an elementary case.

THEOREM 5.2 *In a robotic cell with parallel machines, if* $p_i \leq \delta, \forall i$, *then the forward 1-unit cycle* π_U *achieves the optimum per unit cycle time* $T(\pi_U)$.

Proof. We have previously seen that $T(\pi_U) = 2(m+1)\epsilon + \sum_{i=1}^{m} p_i + (m+2)\delta$ (Chapter 3). Its optimality follows from Theorem 5.1. ∎

COROLLARY 5.1 *If* $p_i \leq \delta, \forall i$, *there is no benefit to be gained from using parallel machines.*

Proof. $p_i \leq \delta, \forall i$, implies that π_U achieves the lower bound on the optimum per unit cycle time as stated in Theorem 5.1. Therefore, the per unit cycle time cannot be improved by adding parallel machines. ∎

5.1.2.2 Blocked Cycles

For $k \geq 2$, the number of k-unit cycles for an m-stage simple robotic cell is much greater than $m!$, the number of 1-unit ($k = 1$) cycles. For example, in a simple robotic cell, if $m = 3$ and $k = 2$, there are 20 cycles. For $m = 4$ and $k = 2$, there are 260 cycles. For a robotic cell with parallel machines, the number of distinct cycles is even larger. Therefore, we narrow our field of study to a particular subset of k-unit cycles called *blocked cycles*. We will define blocked cycles, derive an expression for the cycle time of a general blocked cycle, and then characterize a dominating subset of blocked cycles.

Blocked cycles form a highly structured subclass of k-unit cycles and are a natural generalization of 1-unit cycles. A blocked cycle is composed

of k blocks of activities. Each block has $m+1$ activities, one for each stage $0, 1, 2, \ldots, m$. Therefore, in each block, one machine in each stage $0, \ldots, m$, is unloaded, one machine in each stage $1, \ldots, m+1$, is loaded. For a given cycle, each block has the same order of the activities by numbers, i.e., each block unloads the stages in the same order. This ordering is called the *base permutation*. The letters of the activities, which represent specific machines at each stage, change from block to block to indicate the loading and unloading of different machines; they are restricted only by feasibility.

EXAMPLE 5.2 For example, in a cell with two stages ($m = 2$) with $m_1 = 2$ and $m_2 = 3$, consider the following six-unit blocked cycle with base permutation $(0, 1, 2)$:

$$\begin{aligned}\pi_2 = \quad & (A_{0*a}, A_{1bc}, A_{2a*}, A_{0*b}, A_{1ab}, A_{2c*}, A_{0*a}, A_{1bc}, A_{2b*}, \\ & A_{0*b}, A_{1aa}, A_{2c*}, A_{0*a}, A_{1bb}, A_{2a*}, A_{0*b}, A_{1aa}, A_{2b*}).\end{aligned}$$

At the beginning of each iteration of this cycle, there are parts being processed on machines M_{1b} and M_{2a}. In the first block, the robot unloads a part from I and carries it to M_{1a}. It then travels to M_{1b}, unloads it, and transfers the part to M_{2c}. Next, it moves to M_{2a}, from which it unloads a completed part that is taken to the output buffer.

The robot starts the second block after returning to I. It obtains a raw part and loads it onto M_{1b}. The robot then unloads M_{1a} and carries that part to M_{2b}. This block concludes when the robot unloads a completed part from M_{2c} and places it into O. Processing for the remaining four blocks is similar.

Note that in each block, the indices of the activities follow the $(0, 1, 2)$ permutation, whereas the letters change to indicate the use of different machines. In addition, many activities occur more than once during a cycle, so machines are generally used more than once per cycle, e.g., A_{0*a} and A_{0*b} are each performed three times and A_{1bc} is performed twice. M_{2b} is loaded twice: once by A_{1ab} and once by A_{1bb}. A Gantt chart of this cycle can be found in Figure 5.3.

Most machines $M_{i\ell}$ are loaded, process a part, and are unloaded more than once during a cycle. Each occurrence of this sequence is called a *usage* of $M_{i\ell}$. In cycle π_2 of Example 5.2, there are three usages each of M_{1a} and M_{1b}. If $\Gamma_{i\ell}$ is the number of usages per cycle for machine

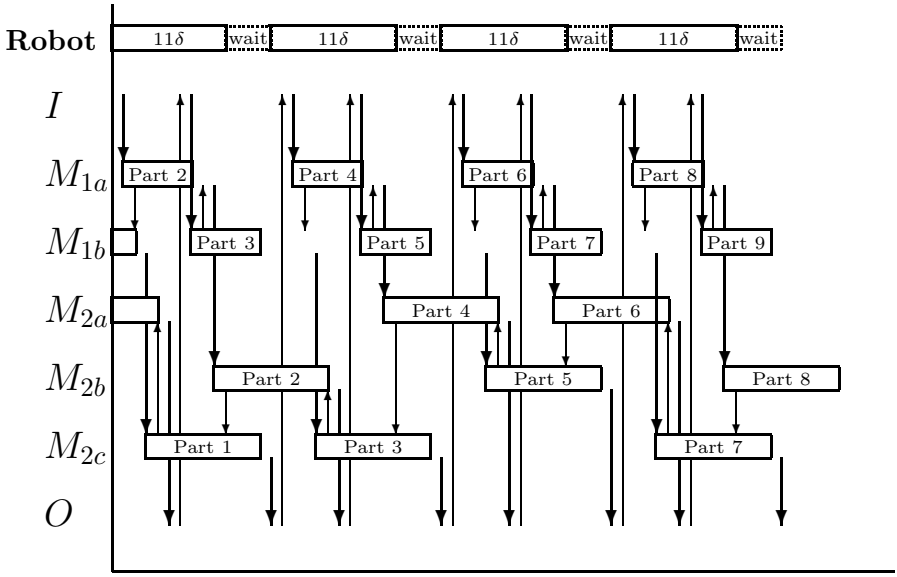


Figure 5.3. Gantt Chart for Blocked Cycle π_2 of Example 5.2. Thin Lines Indicate that the Robot is Traveling without a Part. In this Cell, $p_1 = 30$, $p_2 = 50$, $\delta = 5$, and $\epsilon = 0$.

$M_{i\ell}$, then for each stage $i \in M$, $\sum_{\ell=a}^{\alpha(m_i)} \Gamma_{i\ell} = k$: some machine M_{ij} in stage i is loaded in each block and some machine $M_{i\ell}$ is unloaded in each block (j and ℓ may or may not be equal). The rth usage of $M_{i\ell}$ is denoted $M_{i\ell}^r$. $M_{i\ell}^1$ begins with the first loading of $M_{i\ell}$. This happens at the first occurrence of $A_{i-1,x\ell}$, $x \in \{a, \ldots, \alpha(m_{i-1})\}$.

The existence of a stage s for which some usage $M_{s\beta}^q$ has full waiting (defined in Chapter 3) implies that the subsequence $(s-1, s)$ is in the base permutation. Furthermore, $m_s = 1$. If the base permutation contains a subsequence $(s-1, s)$ and $m_s > 1$, it will be assumed that the robot's cycle does not contain a subsequence $(A_{s-1,\alpha\beta},\ A_{s\beta\gamma})$ in which the robot has full waiting at $M_{s\beta}$. If the robot were to do this, there would be no time advantage gained from having parallel machines at stage s, since another machine in stage s could not be processing at the same time that $M_{s\beta}$ did. Thus, in a blocked cycle, if stage s has a machine with full waiting at some usage, it has only one machine (denoted

M_{sa}), and that machine has full waiting in each of its $\Gamma_{sa} = k$ usages. The set of indices of stages whose machines have full waiting is V_1.

If machine usage $M^r_{i\ell}$ has partial waiting, then its waiting time is denoted by $w^r_{i\ell}$. From the previous paragraph, we see that in a blocked cycle, if the usage $M^r_{i\ell}$ has partial waiting, then all usages of all machines of stage i have partial waiting. The set of indices of stages whose machines have partial waiting is denoted V_2.

Cycle Time for Blocked Cycles. For any cycle π, its cycle time $T(\pi)$ is the sum of the robot's total move time (t_m), the total load/unload time (t_l), the total time for full waiting (W_f), and the total time for partial waiting (W_p). It is easy to see that for a k-unit blocked cycle in a cell with constant travel times, $t_m + t_l + W_f = 2k(m+1)(\delta+\epsilon) + k\sum_{i\in V_1}(p_i - \delta)$. Note that the value for this expression depends only on the base permutation, the cell data, and the number of units produced. For a general blocked cycle π,

$$T(\pi) \;=\; 2k(m+1)(\delta+\epsilon) + k\sum_{i\in V_1}(p_i - \delta) + W_p, \tag{5.1}$$

where

$$W_p \;=\; \sum_{i\in V_2}\sum_{\ell=a}^{\alpha(m_i)}\sum_{r=1}^{\Gamma_{i\ell}} w^r_{i\ell}. \tag{5.2}$$

Given a cell, a base permutation, and k, the cycle time $T(\pi)$ is minimized by minimizing W_p. Note that W_p can be modified by specifying π. In other words, we may choose which machines of each stage are loaded and unloaded in each block. To evaluate such choices, we now derive an expression for the partial waiting time at a machine usage. We will later use this to prove a theorem that provides two sufficient conditions for dominant cycles.

To be able to compute the robot's partial waiting time at a machine usage $M^r_{i\ell}$, we must know which activities are performed during $M^r_{i\ell}$'s processing.

DEFINITION 5.2 Activities that are executed between the loading and the unloading of $M^r_{i\ell}$ are called *intervening activities.*

Let $G^r_{i\ell}$ be the set of intervening activities for $M^r_{i\ell}$. In Example 5.2 (see below), $G^1_{2b} = \{A^1_{2c*}, A^2_{0*a}, A^2_{1bc}\}$ and $G^2_{2b} = \{A^2_{2a*}, A^3_{0*b}, A^2_{1aa}\}$. The superscripts designate different instances of the same activity.

EXAMPLE 5.2, CONTINUED:

$$\pi_2 = (A_{0*a}, A_{1bc}, A_{2a*}, A_{0*b}, \underbrace{A_{1ab}}_{\text{load}M^1_{2b}}, \overbrace{A_{2c*}, A_{0*a}, A_{1bc}}^{G^1_{2b}}, \underbrace{A_{2b*}}_{\text{unload}M^1_{2b}},$$

$$A_{0*b}, A_{1aa}, A_{2c*}, A_{0*a}, \underbrace{A_{1bb}}_{\text{load}M^2_{2b}}, \overbrace{A_{2a*}, A_{0*b}, A_{1aa}}^{G^2_{2b}}, \underbrace{A_{2b*}}_{\text{unload}M^2_{2b}}).$$

For any blocked cycle, the partial waiting time at any usage $M^r_{i\ell}$ is

$$w^r_{i\ell} = \max\left\{0, p_i - 2|G^r_{i\ell}|(\delta+\epsilon) - \delta - \sum_{j\in G^r_{i\ell}(V_1)}(p_j - \delta) - \sum_{j\in G^r_{i\ell}(V_2)} w_j\right\}, \quad (5.3)$$

where $G^r_{i\ell}(V_1)$ is the set of activities in $G^r_{i\ell}$ that unload machine usages with full waiting. More explicitly, $G^r_{i\ell}(V_1) = \{A_{jaq} \in G^r_{i\ell} | j \in V_1\}$. Similarly, $G^r_{i\ell}(V_2) = \{A_{jxy} \in G^r_{i\ell} | j \in V_2\}$. In Example 5.2, $V_1 = \emptyset$ and $w^1_{2b} = \max\{0, p_2 - 7\delta - 6\epsilon - w^1_{2c} - w^1_{1b}\}$.

We can now see that the robot's total partial waiting time between the loading of M^1_{2b} by activity A_{1ab} and the unloading of M^1_{2b} by activity A^1_{2b*} is

$$\begin{aligned} w^1_{2c} + w^1_{1b} + w^1_{2b} &= \max\{w^1_{2c} + w^1_{1b}, p_2 - 7\delta - 6\epsilon\} \\ &= \max\left\{\sum_{j\in G^1_{2b}(V_2)} w_j, \widehat{w^1_{2b}}\right\}, \end{aligned}$$

where $\widehat{w^r_{i\ell}} = p_i - 2|G^r_{i\ell}|(\delta+\epsilon) - \delta - \sum_{j\in G^r_{i\ell}(V_1)}(p_j - \delta)$.

Dominant Blocked Cycles. Before presenting our dominance result, we first define two conditions on blocked cycles and prove two lemmas concerning cycles that satisfy these conditions. We then show that the conditions are sufficient for a cycle to be dominant over other cycles with the same base permutation. The two conditions are:

C5.1. Each machine is loaded as soon as possible after it is unloaded, as allowed by the base permutation, i.e., for all $i \in M$ and all $\theta = a, \ldots, \alpha(m_i)$, machine $M_{i\theta}$ is unloaded during activity $A_{i\theta\gamma}$, where $\gamma \in \{a, \ldots, \alpha(m_{i+1})\}$. Because this is a blocked cycle, exactly one

machine in stage i will be loaded within the next m activities. This machine will be $M_{i\theta}$.

C5.2. For each stage i, $i \in M$, each of its machines has the same number of activities between its loading and its unloading for each usage: for each i, $|G^r_{i\ell}| = g_i, \forall \ell, \forall r$.

LEMMA 5.1 *Consider any k-unit blocked cycle π that satisfies Conditions C5.1 and C5.2. For each stage i, $i \in M$, each of its machines is used the same number of times per cycle. This implies that k is a multiple of m_i, $i \in M$, so $\Gamma_{i\ell} = \frac{k}{m_i}, \forall \ell$. Furthermore, these conditions imply that the machines at stage i are loaded cyclically and that they are unloaded in the same order.*

Proof. For any stage i, Condition C5.1 implies that the number of activities between unloading $M^r_{i\ell}$ and loading $M^{r+1}_{i\ell}$ is fixed by the base permutation, for $\ell = a, \ldots, \alpha(m_i), r = 1, \ldots, \Gamma_{i\ell}$ ($\Gamma_{i\ell}+1$ is taken to be 1). Call this number q_i. Condition C5.2 says that the number of activities between loading $M^r_{i\ell}$ and unloading $M^r_{i\ell}$ is g_i, for $\ell = a, \ldots, \alpha(m_i), r = 1, \ldots, \Gamma_{i\ell}$. The number of activities in cycle π is

$$k(m+1) = \Gamma_{i\ell}(q_i + g_i + 2), \forall \ell. \tag{5.4}$$

Hence, $\Gamma_{i\ell}$ is independent of $\ell, \forall i$, so each machine in stage i is used the same number of times per cycle. Furthermore, $\sum_{\ell=a}^{\alpha(m_i)} \Gamma_{i\ell} = k \Rightarrow m_i\Gamma_{i\ell} = k \Leftrightarrow \Gamma_{i\ell} = \frac{k}{m_i}, \forall i, \forall \ell$. Therefore, k must be a multiple of $m_i, \forall i$.

Equation (5.4) implies that

$$\begin{aligned} k(m+1) &= \frac{k}{m_i}(q_i + g_i + 2), \text{ so} \\ g_i &= m_i(m+1) - (q_i + 2). \end{aligned}$$

Condition C5.1 implies that $0 \le q_i \le m-1$, so

$$(m_i - 1)(m+1) \le g_i \le m_i(m+1) - 2.$$

Therefore, since each block contains $m+1$ activities, some activity $(A_{i-1,xy})$ that loads a machine in stage i occurs $m_i - 1$ times between any loading of a machine $M_{i\ell}$ and its corresponding unloading. Hence, each of the $m_i - 1$ other machines of stage i is loaded between successive loadings of a specified $M_{i\ell}$. By Condition C5.2, each is unloaded after

the same number of activities and, therefore, in the same order. Condition C5.1 implies that this order is maintained throughout the cycle. ■

LEMMA 5.2 *Any cycle that satisfies Conditions C5.1 and C5.2 is feasible.*

Proof. Recall that there are two conditions (Chapter 3) that must be satisfied for a cycle to be feasible. These can now be stated and verified as follows:

- Usage $M_{i\ell}^{r+1}$ cannot be loaded unless usage $M_{i\ell}^{r}$ has been unloaded ($\Gamma_{i\ell}+1$ is taken to be 1).

- Usage $M_{i\ell}^{r+1}$ cannot be unloaded before it is loaded.

Since the machines of stage i are loaded cyclically, there are $m_i(m+1)$ activities between $M_{i\ell}^{r}$'s loading and $M_{i\ell}^{r+1}$'s loading. In the proof of Lemma 5.1, we saw that the number of activities between $M_{i\ell}^{r}$'s loading and its unloading is $g_i \leq m_i(m+1)-2$. Therefore, usage $M_{i\ell}^{r}$ is unloaded before the robot tries to load usage $M_{i\ell}^{r+1}$, $\forall \ell$, $\forall r$.

Because this is a blocked cycle, the number of activities between the unloading of $M_{i\ell}^{r}$ and the unloading of $M_{i\ell}^{r+1}$ is a positive multiple of $m+1$. By Condition C5.1, after usage $M_{i\ell}^{r}$ is unloaded, usage $M_{i\ell}^{r+1}$ is loaded within the next m activities. Therefore, $M_{i\ell}^{r+1}$ will have been loaded before the robot attempts to unload it. ■

We are now ready to state and prove a dominance theorem. We prove that for a given base permutation, a cycle that satisfies Conditions C5.1 and C5.2 is dominant over all other blocked cycles. It follows that the sub-class of blocked cycles satisfying Conditions C5.1 and C5.2 dominates the class of blocked cycles.

THEOREM 5.3 *Given a robotic cell with m stages, fixed data (p_i, m_i, $i \in M$; δ, ϵ), and a base permutation, a blocked cycle that satisfies Conditions C5.1 and C5.2 is dominant.*

Proof. We show that $W_p^* = \min\{W_p\}$, where for a given robotic cell, W_p^* is the total partial waiting in a cycle that satisfies Conditions C5.1 and C5.2, and the minimum is taken over all feasible blocked cycles for a given base permutation in that cell.

Note that for any cycle, equation (5.3) can be stated as

$$w_{i\ell}^r = \max\{0, \widehat{w_{i\ell}^r} - \sum_{j \in G_{i\ell}^r(V_2)} w_j\}, \ \forall i, \ell, r. \tag{5.5}$$

Hence, for a k-unit blocked cycle, we have a system of $k|V_2|$ inequalities

$$w_{i\ell}^r + \sum_{j \in G_{i\ell}^r(V_2)} w_j \geq \widehat{w_{i\ell}^r}, \ i \in V_2; \ell = a, \ldots, \alpha(m_i); r = 1, \ldots, \Gamma_{i\ell}, \tag{5.6}$$

where $w_{i\ell}^r \geq 0$. Furthermore, for each machine $M_{i\ell}$, $G_{i\ell}^r \cap G_{i\ell}^q = \emptyset$, for $1 \leq r < q \leq \Gamma_{i\ell}$. Therefore, for each $M_{i\ell}$, $i \in V_2$, we can add the inequalities in (5.6) corresponding to $w_{i\ell}^r$, $r = 1, \ldots, \Gamma_{i\ell}$, and derive the following $\sum_{i \in V_2} m_i$ inequalities:

$$\sum_{r=1}^{\Gamma_{i\ell}} \left[w_{i\ell}^r + \sum_{j \in G_{i\ell}^r(V_2)} w_j \right] \geq \sum_{r=1}^{\Gamma_{i\ell}} \widehat{w_{i\ell}^r}, \quad i \in V_2; \ell = a, \ldots, \alpha(m_i).$$

Claim 1. For a fixed $\Gamma_{i\ell}$, $\sum_{r=1}^{\Gamma_{i\ell}} \widehat{w_{i\ell}^r}$ is minimized by a cycle satisfying Condition C5.1, $\forall i, \ell$.

Proof of Claim 1. Because Condition C5.1 requires that $M_{i\ell}$ be reloaded as soon as possible, it maximizes the sum of the number of activities between the loading and unloading of the usages of a machine; i.e., it maximizes $\sum_{r=1}^{\Gamma_{i\ell}} |G_{i\ell}^r|$. Thus, it also minimizes $\sum_{r=1}^{\Gamma_{i\ell}} \widehat{w_{i\ell}^r}$. □

Claim 2. $\sum_{\ell=a}^{\alpha(m_i)} \sum_{r=1}^{\Gamma_{i\ell}} \max\{0, \widehat{w_{i\ell}^r}\}$ is minimized by a cycle satisfying Conditions C5.1 and C5.2, $\forall i$.

Proof of Claim 2. Condition C5.2 ($|G_{i\ell}^r| = g_i, \forall \ell, \forall r$) implies that $\widehat{w_{i\ell}^r}$ is the same for all usages of all machines in a given stage i. Let $\widehat{w_{i\ell}^r} = \widehat{w_i^*}$ for a cycle satisfying Conditions C5.1 and C5.2. Therefore, since $\sum_{\ell=a}^{\alpha(m_i)} \Gamma_{i\ell} = k$,

$$\sum_{\ell=a}^{\alpha(m_i)} \sum_{r=1}^{\Gamma_{i\ell}} \max\{0, \widehat{w_i^*}\} = \max\left\{0, \sum_{\ell=a}^{\alpha(m_i)} \sum_{r=1}^{\Gamma_{i\ell}} \widehat{w_i^*}\right\}$$

$$= \max\left\{0, k[p_i - 2g_i(\delta+\epsilon) - \delta - \sum_{j \in G_{i\ell}^r(V_1)} (p_j - \delta)]\right\}$$

$$\leq \max\left\{0, \sum_{\ell=a}^{\alpha(m_i)} \sum_{r=1}^{\Gamma_{i\ell}} [p_i - 2|G_{i\ell}^r|(\delta+\epsilon) - \delta - \sum_{j \in G_{i\ell}^r(V_1)} (p_j - \delta)]\right\}$$

$$\leq \sum_{\ell=a}^{\alpha(m_i)} \sum_{r=1}^{\Gamma_{i\ell}} \max\{0, \widehat{w_{i\ell}^r}\}.$$

□

Let $w_{i\ell}^{r*}$ be the waiting time of the robot at machine usage $M_{i\ell}^r$ in a cycle that satisfies Conditions C5.1 and C5.2. If $w_{i\ell}^{r*} = 0$, then obviously $w_{i\ell}^{r*} = \min\{w_{i\ell}^r\}$. If $w_{i\ell}^{r*} > 0$, then equation (5.5) implies

$$w_{i\ell}^{r*} + \sum_{j \in G_{i\ell}^r(V_2)} w_j^* = \widehat{w_i^*}, \ \forall i, \ell, r.$$

Hence, by Claim 2,

$$\begin{aligned}
\sum_{\ell=a}^{\alpha(m_i)} \sum_{r=1}^{\Gamma_{i\ell}} \left[w_{i\ell}^{r*} + \sum_{j \in G_{i\ell}^r(V_2)} w_j^*\right] &= \sum_{\ell=a}^{\alpha(m_i)} \sum_{r=1}^{\Gamma_{i\ell}} \widehat{w_i^*} \\
&\leq \sum_{\ell=a}^{\alpha(m_i)} \sum_{r=1}^{\Gamma_{i\ell}} \max\{0, \widehat{w_{i\ell}^r}\} \\
&\leq \sum_{\ell=a}^{\alpha(m_i)} \sum_{r=1}^{\Gamma_{i\ell}} \left[w_{i\ell}^r + \sum_{j \in G_{i\ell}^r(V_2)} w_j\right], \ \forall i.
\end{aligned}$$

This argument holds for all stages $i \in V_2$. Therefore, $W_p^* = \min\{W_p\}$. ■

5.1.3 LCM Cycles

As we have previously seen (Chapter 3), when dealing with multi-unit cycles, one must address their feasibility. In this section, we define LCM (for *least common multiple*) cycles that satisfy Conditions C5.1 and C5.2 stated in Section 5.1.2.2, and, therefore, are always feasible. This led

Kumar et al. [102] to use them in their genetic algorithm-based analysis of a specific company's robotic cell. They also show that LCM cycles greatly increase throughput in that cell. We now consider LCM cycles for general cells with parallel machines.

To motivate LCM cycles, note that as proven in Lemma 5.1, k must be a multiple of each m_i for Theorem 5.3 to be fulfilled. This leads to the definition of LCM cycles.

DEFINITION 5.3 *LCM Cycles* are blocked cycles that have the following characteristics:

- They satisfy Conditions C5.1 and C5.2 stated in Section 5.1.2.2.
- The number of blocks is $\lambda = LCM[m_1, m_2, \ldots, m_m]$.
- For each stage i, the loading of its machines is ordered alphabetically, beginning with machine M_{ia} in the first block.

The last requirement ensures that a cell has a unique LCM cycle for a given base permutation. From Theorem 5.3, we know that for a given base permutation, the LCM cycle dominates all other blocked cycles. Therefore, the class of LCM cycles dominates the class of blocked cycles.

In the remainder of this section, we further characterize LCM cycles and derive results. This includes providing examples and computing the cycle times for those examples.

EXAMPLE 5.3 Consider a two-stage cell ($m = 2$) in which $m_1 = 2$ and $m_2 = 3$. The LCM cycle for base permutation $(0, 2, 1)$ is

$$\begin{aligned}\pi_{LD}(2,3) \quad = \quad & (A_{0*a}, A_{2a*}, A_{1ba}, A_{0*b}, A_{2b*}, A_{1ab}, A_{0*a}, A_{2c*}, A_{1bc}, \\ & A_{0*b}, A_{2a*}, A_{1aa}, A_{0*a}, A_{2b*}, A_{1bb}, A_{0*b}, A_{2c*}, A_{1ac}).\end{aligned}$$

At the beginning of each iteration of this cycle, there are parts being processed on machines M_{1b}, M_{2a}, M_{2b}, and M_{2c}. In the first block, the robot unloads a part from I and carries it to M_{1a}. Next, it moves to M_{2a}, from which it unloads a completed part, which it takes to the output buffer. It then travels to M_{1b}, unloads it, and transfers the part to M_{2a}.

The robot starts the second block after returning to I. It obtains a raw part and loads it onto M_{1b}. The robot then unloads a completed part from M_{2b} and places it into O. This block concludes when the robot

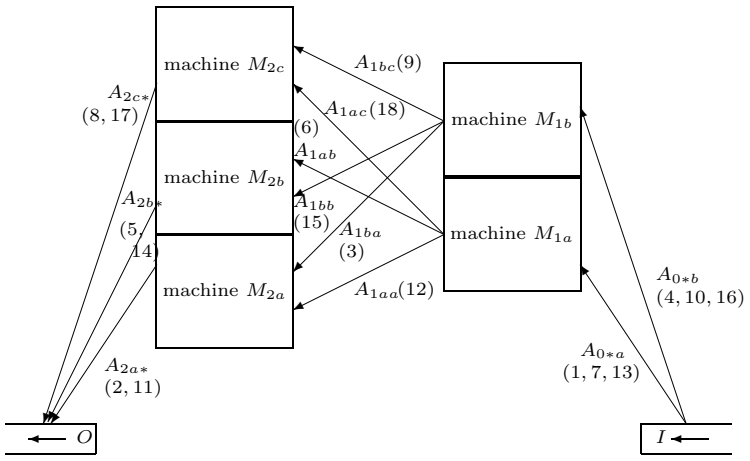


Figure 5.4. LCM Cycle $\pi_{LD}(2,3)$. Numbers in Parentheses Indicate Order of Operations.

unloads M_{1a} and carries that part to M_{2b}. Processing for the remaining four blocks is similar.

We use π_{LD} to denote the general LCM cycle that has π_D as its base permutation. The specific instance in Example 5.3 is denoted $\pi_{LD}(2,3)$ because $m_1 = 2$ and $m_2 = 3$. It has six ($LCM[2,3]$) blocks of three ($m+1$) activities each. Each block is ordered $A_0A_2A_1$. For a given stage, we rotate usage of its machines, beginning with loading machine M_{ia} for each. For example, the blocks alternate between loading M_{1a} (A_{0*a}) and M_{1b} (A_{0*b}) – each is used $\lambda/m_1 = 3$ times per cycle (Lemma 5.1). Each of the three machines in stage 2 is loaded every third block – twice per cycle. Each usage of a machine in stage 1 has four activities between its loading and its unloading; each in stage 2 has seven (Condition C5.2 of Section 5.1.2.2). A schematic representation of this cycle is presented in Figure 5.4. A Gantt chart of $\pi_{LD}(2,3)$ can be found in Figure 5.5.

Note that π_{LD} satisfies Condition C5.1 by loading each machine as soon as possible after unloading it. Since processing is cyclical, minimizing the time between unloading and loading each machine maximizes the time between loading and returning to unload each machine. This maximization in turn minimizes the robot's waiting time at each machine usage.

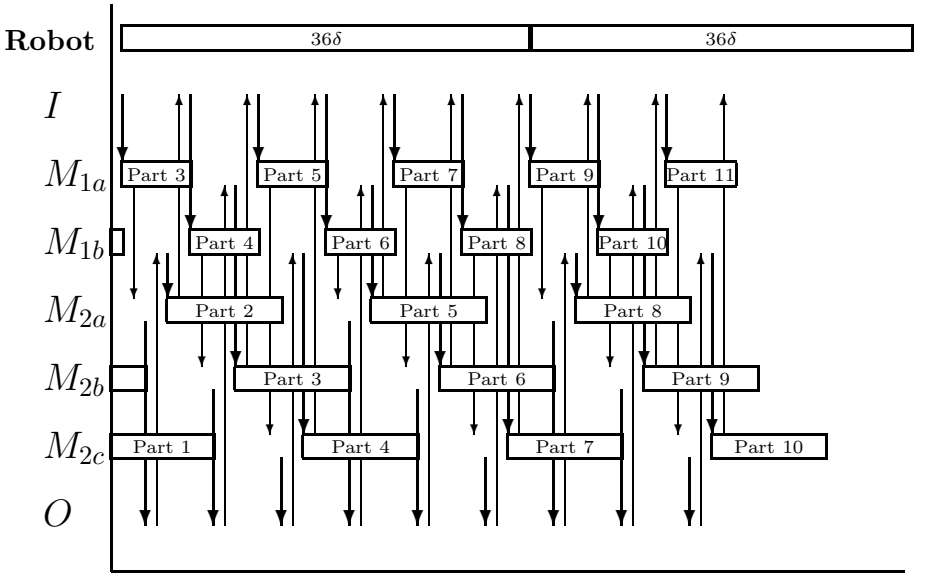


Figure 5.5. Gantt Chart for Cycle $\pi_{LD}(2,3)$. Thin Lines Indicate that the Robot is Traveling Without a Part. In this Cell, $p_1 = 30$, $p_2 = 50$, $\delta = 5$, and $\epsilon = 0$.

The cycle time of the general cycle π_{LD} can be easily computed from equations (5.1) and (5.3). Since $V_1 = \emptyset$ in π_{LD}, we have

$$T(\pi_{LD}) = 2\lambda(m+1)(\delta+\epsilon) + \max_{1\le i\le m}\left\{\max\left\{0, \frac{\lambda}{m_i}[p_i - 2g_i(\delta+\epsilon) - \delta]\right\}\right\}.$$

Since $g_i = m_i(m+1) - 2,\ \forall i$, this leads to

$$T(\pi_{LD}) = \max\left\{2\lambda(m+1)(\delta+\epsilon), \max_{1\le i\le m}\left\{\frac{\lambda}{m_i}(p_i + 3\delta + 4\epsilon)\right\}\right\}. \quad (5.7)$$

Therefore,

$$T(\pi_{LD}(2,3)) = \max\{36\delta + 36\epsilon, 3(p_1 + 3\delta + 4\epsilon), 2(p_2 + 3\delta + 4\epsilon)\}.$$

The LCM cycle in this cell based on π_U or, equivalently, with base permutation $(0,1,2)$, is

$$\begin{aligned}\pi_{LU}(2,3) \quad = \quad & (A_{0*a}, A_{1ba}, A_{2b*}, A_{0*b}, A_{1ab}, A_{2c*}, A_{0*a}, A_{1bc}, A_{2a*},\\ & A_{0*b}, A_{1aa}, A_{2b*}, A_{0*a}, A_{1bb}, A_{2c*}, A_{0*b}, A_{1ac}, A_{2a*}).\end{aligned}$$

A similar computation yields the cycle time of π_{LU} as

$$T(\pi_{LU}(2,3)) = \max\{36\delta + 36\epsilon, 3(p_1 + 5\delta + 6\epsilon), 2(p_2 + 5\delta + 6\epsilon)\}.$$

Note that if the robot never waits, $T(\pi_{LU}(2,3)) = T(\pi_{LD}(2,3))$. It is when the robot must wait that $\pi_{LD}(2,3)$ proves itself as the better cycle.

5.1.4 Practical Implications

In this section, we state and prove a theorem that specifies an optimal cycle for a very common special case. This theorem is used to generate a formula for determining how many machines are needed at each stage to meet a given throughput requirement.

5.1.4.1 Optimal Cycle for a Common Case

We now prove that π_{LD} is an optimal cycle if $p_i \geq \delta, \forall i$. Our work with a Dallas-area semiconductor equipment manufacturer [64] supports that this is common in practice, as do the studies [102] and [128]. Note that the following theorem produces an optimal cycle over *all* cycles, not just blocked cycles. Its proof is a straightforward extension, for parallel machines, of the argument for the optimality of the reverse cycle π_D in simple cells under the assumption $p_i \geq \delta, \forall i$ (see Chapter 3).

THEOREM 5.4 *If $p_i \geq \delta, \forall i$, then π_{LD} achieves the optimum per unit cycle time.*

Proof. By Theorem 5.1, if $p_i \geq \delta, \forall i$, then $T(\pi) \geq 2k(m+1)\epsilon + km\delta + k(m+2)\delta = 2k(m+1)(\delta+\epsilon)$, for all k-unit cycles π. Hence, $T(\pi)/k \geq 2(m+1)(\delta+\epsilon)$. From equation (5.7), for the λ-unit cycle π_{LD}, $T(\pi_{LD}) = 2\lambda(m+1)(\delta+\epsilon)$ if the robot never has to wait at a machine for it to complete processing, so its per unit cycle time in this case is $2(m+1)(\delta+\epsilon)$.

Observe that the robot's fewest actions after a specific machine $M_{i\ell}$ completes processing and before that machine starts processing its next part are (1) unload $M_{i\ell}$, (2) travel to a machine in stage $i+1$, say $M_{i+1,a}$, (3) load $M_{i+1,a}$, (4) travel to a machine in stage $i-1$, say $M_{i-1,b}$, (5) unload $M_{i-1,b}$, (6) travel to $M_{i\ell}$, and (7) load $M_{i\ell}$, for a total minimum possible time between the unloading and the loading of $M_{i\ell}$ of $3\delta + 4\epsilon$. Thus, the minimum possible time between each loading of $M_{i\ell}$ is $p_i + 3\delta + 4\epsilon$. Since $M_{i\ell}$ is used $\Gamma_{i\ell}$ times per cycle, the minimum possible time

required for its total processing during a complete cycle is $\Gamma_{i\ell}(p_i+3\delta+4\epsilon)$. The best possible total cycle time is

$$\max_{1\le i\le m}\left\{\max_{a\le \ell\le \alpha(m_i)}\{\Gamma_{i\ell}(p_i+3\delta+4\epsilon)\}\right\}.$$

Since $\sum_{\ell=a}^{\alpha(m_i)}\Gamma_{i\ell}=k,\forall i$, we minimize $\max_{a\le\ell\le\alpha(m_i)}\{\Gamma_{i\ell}(p_i+3\delta+4\epsilon)\}$ by setting $\Gamma_{i\ell}=k/m_i,\forall i,\forall\ell$. Therefore, the best possible cycle time is $\max_{1\le i\le m}\{(k/m_i)(p_i+3\delta+4\epsilon)\}$. Since $T(\pi_{LD})=\max\{2\lambda(m+1)(\delta+\epsilon),\max_{1\le i\le m}\{(\lambda/m_i)(p_i+3\delta+4\epsilon)\}\},\pi_{LD}$ achieves the optimum cycle time for a λ-unit cycle. The requirement $\Gamma_{i\ell}=k/m_i,\forall i$, only makes sense if k is a multiple of $m_i,\forall i$. Therefore, π_{LD} achieves the optimum per unit cycle time over all k-unit cycles, $k\ge 1$. ∎

COROLLARY 5.2 *If* $\max_{1\le i\le m}\{\frac{\lambda}{m_i}(p_i+3\delta+4\epsilon)\}\ge 2\lambda(m+1)(\delta+\epsilon)$, *then* π_{LD} *achieves the optimum per unit cycle time.*

Proof. Follows directly from the proof of Theorem 5.4. ∎

COROLLARY 5.3 *Suppose* π_{LD} *is optimal. For any stage* $i\in M$ *for which* $p_i\le(2m-1)\delta+2(m-1)\epsilon$, *there is no benefit to be gained from using parallel machines in stage* i.

Proof. Observe that

$$p_i\le(2m-1)\delta+2(m-1)\epsilon \iff p_i+3\delta+4\epsilon\le 2(m+1)(\delta+\epsilon).$$

Therefore, adding parallel machines to stage i cannot reduce $T(\pi_{LD})$. ∎

Consider the alternative case in which $p_i\ge\delta,\ \forall i$, is not true. In much the same way that π_D is a 2-approximation for an optimal cycle in a simple cell (Chapter 3), π_{LD} provides a 2-approximation for an optimal cycle in a robotic cell with parallel machines. From Theorem 5.1 and the proof of Theorem 5.4, we know that for any cell, independent of the relationship between $p_i,\ i\in M$, and δ, for an optimal cycle π^*,

$$\max\left\{2\lambda(m+1)\epsilon+\lambda\sum_{i=1}^{m}\min\{p_i,\delta\}+\lambda(m+2)\delta,\ \max_{1\le i\le m}\left\{\frac{\lambda}{m_i}(p_i+3\delta+4\epsilon)\right\}\right\}\le T(\pi^*).$$

From equation (5.7), we have

$$T(\pi_{LD}) - \lambda m\delta \leq \max\left\{2\lambda(m+1)\epsilon + \lambda\sum_{i=1}^{m}\min\{p_i,\delta\} + \lambda(m+2)\delta, \max_{1\leq i\leq m}\left\{\frac{\lambda}{m_i}(p_i+3\delta+4\epsilon)\right\}\right\}.$$

Hence, $T(\pi_{LD}) \leq T(\pi^*) + \lambda m\delta \leq 2T(\pi^*)$.

5.1.4.2 Fewest Machines Required to Meet Timelines

The previous theorem leads to a guideline for adding machines to meet mandated throughput requirements if $p_i \geq \delta, \forall i$, which we have seen is an important practical case. If a customer specifies a throughput that implies the per unit cycle time T^*, then we must have $T(\pi_{LD})/\lambda \leq T^*$. Therefore, for each stage i, we must have $(p_i + 3\delta + 4\epsilon)/m_i \leq T^*$. (Note that if $2(m+1)(\delta+\epsilon) > T^*$, then this time requirement cannot be satisfied). To meet the time requirement, the number of parallel machines for each stage i must satisfy $(p_i + 3\delta + 4\epsilon)/T^* \leq m_i$. To minimize overall cost, for each i, choose the smallest such m_i. Therefore,

$$m_i = \left\lceil \frac{p_i + 3\delta + 4\epsilon}{T^*} \right\rceil, \ i = 1, \ldots, m. \tag{5.8}$$

Examples that illustrate the increase in throughput that can be realized by adding parallel machines can be found in Section 5.2.3.2.

In the case with general processing times, π_{LD} may or may not be an optimal cycle. However, the preceding analysis is still valid, so if equation (5.8) is satisfied and $2(m+1)(\delta+\epsilon) \leq T^*$, then π_{LD} will meet the required timeline.

5.2 Dual-Gripper Robots

For dual-gripper cells with parallel machines, we again consider only constant travel-time cells $(RF_m^2(m_1, ..., m_m)|(free, C, cyclic\text{-}k)|\mu)$ [61]. As in the previous section, we provide an optimal solution to the k-unit cycle problem under conditions that are common in practice. The main idea of this analysis is the construction of a specific cycle $\bar{C}_{d,L}^m$, which combines the structures of LCM cycles (Section 5.1.3) and the cycle $\bar{C}_d^m$ for dual gripper simple cells (Chapter 4).

5.2.1 Lower Bound on Per Unit Cycle Time

We now state and prove the lower bound on the per unit cycle time for a dual-gripper robotic cell with parallel machines. Both the bound and its proof are similar to those of Theorems 4.7 and 4.8.

THEOREM 5.5 *If $\theta \leq \delta$ and $p_i \geq \delta$, $i = 1, \ldots, m$, then for any k-unit cycle π in a robotic cell with parallel machines served by a dual-gripper robot, the cycle time $T(\pi)$ satisfies*

$$\frac{T(\pi)}{k} \geq \max\left\{(m+2)\delta + 2(m+1)\epsilon + m\theta, \max_{1\leq i\leq m}\left\{\frac{p_i + 2\epsilon + \theta}{m_i}\right\}\right\} \tag{5.9}$$

Proof. In part A of the proof, we show that $\max_i\{(p_i + 2\epsilon + \theta)/m_i\}$ is a lower bound for $T(\pi)/k$. In part B, we show that $(m+2)\delta + 2(m+1)\epsilon + m\theta$ is a lower bound.

Part A: Suppose we represent a cycle by the sequence

$$(M_{i\ell}^-, \sigma_1, M_{i\ell}^+, \sigma_2, M_{i\ell}^-, \sigma_3, M_{i\ell}^+, \sigma_4, \ldots, M_{i\ell}^-, \sigma_{2\Gamma_{i\ell}-1}, M_{i\ell}^+, \sigma_{2\Gamma_{i\ell}})$$

for some specific machine $M_{i\ell}$. The time between the beginning of $M_{i\ell}^-$ and the end of $M_{i\ell}^+$ is at least $p_i + 2\epsilon$. If $p_i > \delta$, $\forall i$, we can establish a lower bound by choosing $\widehat{\sigma}_2 = \widehat{\sigma}_4 = \cdots = \widehat{\sigma}_{2\Gamma_{i\ell}} = \emptyset$. This leads to

$$T(\pi) \geq \max_{1\leq i\leq m}\max_{a\leq \ell\leq \alpha(i)}\{\Gamma_{i\ell}(p_i + 2\epsilon + \theta)\}.$$

Since $\sum_{\ell=a}^{\alpha(m_i)}\Gamma_{i\ell} = k$, $\forall i$, $\max_{a\leq\ell\leq\alpha(i)}\{\Gamma_{i\ell}\} \geq \frac{k}{m_i}$, so

$$T(\pi) \geq \max_{1\leq i\leq m}\left\{\frac{k}{m_i}(p_i + 2\epsilon + \theta)\right\}.$$

Part B: We again examine the cases detailed in the proof of Theorem 4.8. However, we now consider the robot's residence times at *stage i*, rather than at *machine M_i*. For the sequence $(M_{i\rho}^-, M_{i\beta}^+)$, $\rho \neq \beta$, the residence time is $2\epsilon + \delta$: load $M_{i\rho}$, move to $M_{i\beta}$, unload $M_{i\beta}$. In this analysis, we establish that the lower bound for each case in a simple robotic cell applies to the respective case for cells with parallel machines. Hence, the proof of Theorem 4.8 can be directly applied to prove Theorem 5.5.

For any stage i, any sequence σ of a k-unit cycle that represents two consecutive loadings and unloadings of machines in stage i is of

the form $\sigma = (M^-_{i\rho,2r-1}, \sigma_1, M^+_{i\beta,2r-1}, \sigma_2, M^-_{i\gamma,2r}, \sigma_3, M^+_{i\eta,2r}, \sigma_4)$, where $r = 1, \ldots, k/2$, and $M^-_{i\cdot,2r}$ ($M^+_{i\cdot,2r}$) denotes $2r^{th}$ loading (unloading) of a machine in stage i. Note that in σ at least one of $\widehat{\sigma}_1$, $\widehat{\sigma}_2$, $\widehat{\sigma}_3$, and $\widehat{\sigma}_4$ is not empty.

Case 0. $M_i = M_0$. The analysis is identical to that of Theorem 4.8.

Case 1. $M_i = M_{m+1}$. The analysis is identical to that of Theorem 4.8.

Case 2. $\widehat{\sigma}_1 \neq \emptyset, \widehat{\sigma}_2 \neq \emptyset, \widehat{\sigma}_3 \neq \emptyset, \widehat{\sigma}_4 \neq \emptyset$. The robot is occupied at machines in stage i for at least 4ϵ time units.

Case 3. $\widehat{\sigma}_1 = \emptyset$ (for $\widehat{\sigma}_3 = \emptyset$ the analysis is similar). If $\rho = \beta$, then the robot is occupied at stage i for at least $4\epsilon + p_i$ time units. If $\rho \neq \beta$, then the robot is occupied at stage i for at least $4\epsilon + \delta$ time units. Hence, in this case, the robot's minimum residence time at stage i is $4\epsilon + \delta$.

Case 4. $\widehat{\sigma}_2 = \emptyset$ (for $\widehat{\sigma}_4 = \emptyset$ the analysis is similar). If $\beta = \gamma$, then the robot is occupied at stage i for at least $4\epsilon + \theta$ time units. If $\beta \neq \gamma$, then the robot is occupied at stage i for at least $4\epsilon + \delta$ time units (we assume that the robot switches grippers while moving between machines). Hence, in this case, the robot's minimum residence time at stage i is $4\epsilon + \theta$ time units.

Case 5. $\widehat{\sigma}_1 = \widehat{\sigma}_2 = \emptyset$ (for $\widehat{\sigma}_3 = \widehat{\sigma}_4 = \emptyset$ the analysis is similar). The following subcases imply the listed minimum residence times:

a) $\rho = \beta = \gamma \;\Rightarrow\; 4\epsilon + \theta + p_i$;

b) $\rho = \beta \neq \gamma \;\Rightarrow\; 4\epsilon + \delta + p_i$;

c) $\rho \neq \beta = \gamma \;\Rightarrow\; 4\epsilon + \theta + \delta$;

d) $\rho \neq \beta \neq \gamma \;\Rightarrow\; 4\epsilon + 2\delta$.

Hence, the robot is occupied at stage i for at least $4\epsilon + \theta + \delta$ time units.

Case 6. $\widehat{\sigma}_1 = \widehat{\sigma}_3 = \emptyset$. Recall that "$\wedge$" means *logical and*, and "$\vee$" means *logical or*.

a) $\rho = \beta \wedge \gamma = \eta \;\Rightarrow\; 4\epsilon + 2p_i$;

b) $(\rho = \beta \wedge \gamma \neq \eta) \vee (\rho \neq \beta \wedge \gamma = \eta) \;\Rightarrow\; 4\epsilon + \delta + p_i$;

c) $(\rho \neq \beta \wedge \gamma \neq \eta) \Rightarrow 4\epsilon + 2\delta$.

Hence, the robot is occupied at stage i for at least $4\epsilon + 2\delta$ time units.

Case 7. $\widehat{\sigma}_1 = \widehat{\sigma}_4 = \emptyset$ (for $\widehat{\sigma}_2 = \widehat{\sigma}_3 = \emptyset$ the analysis is similar).

a) $\rho = \beta = \eta \Rightarrow 4\epsilon + \theta + p_i$;

b) $\rho = \beta \neq \eta \Rightarrow 4\epsilon + \delta + p_i$;

c) $\rho = \eta \neq \beta \Rightarrow 4\epsilon + \theta + \delta$;

d) $\eta \neq \rho \neq \beta \Rightarrow 4\epsilon + 2\delta$.

Hence, the robot is occupied at stage i for at least $4\epsilon + \theta + \delta$ time units.

Case 8. $\widehat{\sigma}_2 = \widehat{\sigma}_4 = \emptyset$.

a) $\beta = \gamma \wedge \rho = \eta \Rightarrow 4\epsilon + 2\theta$;

b) $\beta \neq \gamma \wedge \rho \neq \eta \Rightarrow 4\epsilon + 2\delta$;

c) $(\beta = \gamma \wedge \rho \neq \eta) \vee (\beta \neq \gamma \wedge \rho = \eta) \Rightarrow 4\epsilon + \theta + \delta$.

Hence, the robot is occupied at stage i for at least $4\epsilon + 2\theta$ time units.

Case 9. $\widehat{\sigma}_1 = \widehat{\sigma}_2 = \widehat{\sigma}_3 = \emptyset$ (for $\widehat{\sigma}_1 = \widehat{\sigma}_3 = \widehat{\sigma}_4 = \emptyset$ the analysis is similar.)

a) $\rho = \beta = \gamma = \eta \Rightarrow 4\epsilon + \theta + 2p_i$;

b) $\rho = \beta = \gamma \neq \eta \Rightarrow 4\epsilon + \theta + \delta + p_i$;

c) $\rho = \beta \neq \gamma = \eta \Rightarrow 4\epsilon + \delta + 2p_i$;

d) $\rho = \beta \neq \gamma \neq \eta \vee \gamma = \eta \neq \rho \neq \beta \Rightarrow 4\epsilon + 2\delta + p_i$;

e) $\beta = \eta \neq \gamma \vee \rho = \gamma \neq \beta \Rightarrow$ infeasible;

f) $\rho \neq \beta = \gamma = \eta \Rightarrow 4\epsilon + \theta + \delta + p_i$;

g) $\rho \neq \beta \wedge \rho \neq \gamma \wedge \rho \neq \eta \wedge \beta \neq \gamma \wedge \beta \neq \eta \wedge \gamma \neq \eta \Rightarrow 4\epsilon + 3\delta$;

h) $\rho \neq \beta = \gamma \neq \eta \Rightarrow 4\epsilon + \theta + 2\delta$.

Hence, the robot is occupied at stage i for at least $4\epsilon + \theta + 2\delta$ time units.

Case 10. $\widehat{\sigma}_1 = \widehat{\sigma}_2 = \widehat{\sigma}_4 = \emptyset$ (or $\widehat{\sigma}_2 = \widehat{\sigma}_3 = \widehat{\sigma}_4 = \emptyset$). This schedule structure is infeasible.

Again, let $u_j, j = 0, \ldots, 9$, denote the number of sequences σ corresponding to Case j that occur in a k-unit cycle, for all $r = 1, \ldots, k/2$, stages $i = 0, \ldots, m+1$, and machines $M_{i\ell}$, $\ell = a, \ldots, \alpha(m_i)$. As there are $k/2$ sequences for each stage in a k-unit cycle, we have $mk/2 = u_2 + u_3 + u_4 + u_5 + u_6 + u_7 + u_8 + u_9$ and $u_0 = u_1 = k/2$. By adding residence times corresponding to all the above cases and setting $p_i = \delta$, $\forall i$, we get a lower bound for T_r, the aggregate residence time of the robot at all machines in all stages:

$$\begin{aligned} T_r \geq \frac{k}{2}\mu + 2mk\epsilon &+ (u_4 + u_5 + u_7 + 2u_8 + u_9)\theta \\ &+ (u_3 + u_5 + 2u_6 + u_7 + 2u_9)\delta, \end{aligned}$$

where μ denotes the minimum residence time of the robot at both I and O in sequences $(M^+_{0,2r-1}, \sigma_1, M^+_{0,2r}, \sigma_2)$ and $(M^-_{m+1,2r-1}, \sigma_1, M^-_{m+1,2r}, \sigma_2)$. Hence, the lower bound for T_r is the same as the one found in Theorem 4.8.

A robot movement from $M_{i\gamma}$ to $M_{h\eta}$ occurs when an operation $M^-_{i\gamma}$ or $M^+_{i\gamma}$ is followed immediately by a robot operation $M^-_{h\eta}$ or $M^+_{h\eta}$, where $h \neq i$ or $\gamma \neq \eta$. Such a movement is incident to both machines $M_{i\gamma}$ and $M_{h\eta}$, and requires δ time units. Therefore, the proof of Theorem 4.8 can be directly applied to establish the same lower bound for the aggregate travel time T_t, and hence for Case B. This proves the lower bound of inequality (5.9). ∎

5.2.2 An Optimal Cycle

Before defining Cycle $\bar{C}^m_{d,L}$, in order to develop intuition, we first provide an example and then describe the cycle in general. Example 5.4 below presents a specific case of cycle $\bar{C}^m_{d,L}$ in a cell with two stages, where $m_1 = 2$ and $m_2 = 3$. It begins with all machines occupied and the robot at I with both grippers empty. The robot unloads a part from the input buffer and travels to M_{1a}. If necessary, the robot waits at M_{1a} until it completes processing. The robot then unloads that part and replaces it in M_{1a} with the one from I. The part from M_{1a} is then carried to M_{2a}, at which the robot waits (if necessary), unloads the part from M_{2a}, and replaces it with the part from M_{1a}. The part from M_{2a} is carried to and loaded onto O. Then the robot returns to I for the next raw part. The robot again travels to stages 1 and 2, but this time it

serves machines M_{1b} and M_{2b}. In the following pass through the stages, it serves machines M_{1a} and M_{2c}. The robot makes three more passes through the stages, serving machines M_{1b} and M_{2a}, then machines M_{1a} and M_{2b}, and finally machines M_{1b} and M_{2c}. After loading the last part onto O, the robot completes the cycle by returning to I. By cycling through the machines in each stage in this fashion, the robot's waiting time at each machine is minimized.

For a cell with parallel machines, our notation for dual-gripper robots is extended as follows. $R^-_{ia}(0,h)$ or $R^-_{ia}(h,0)$ is the robot state in which the robot has just finished loading a part onto machine M_{ia}. $R^+_{ib}(i+1,h)$ or $R^+_{ib}(h,i+1)$ is the state in which the robot has just finished unloading a part from machine M_{ib}. The subscript $0*$ is used when unloading I, and the subscript $(m+1)*$ is used when loading O.

EXAMPLE 5.4

$$
\begin{aligned}
&(R^+_{0*}(1,0), R^+_{1a}(1,2), R^-_{1a}(0,2), R^+_{2a}(3,2), R^-_{2a}(3,0), R^-_{3*}(0,0),\\
&R^+_{0*}(1,0), R^+_{1b}(1,2), R^-_{1b}(0,2), R^+_{2b}(3,2), R^-_{2b}(3,0), R^-_{3*}(0,0),\\
&R^+_{0*}(1,0), R^+_{1a}(1,2), R^-_{1a}(0,2), R^+_{2c}(3,2), R^-_{2c}(3,0), R^-_{3*}(0,0),\\
&R^+_{0*}(1,0), R^+_{1b}(1,2), R^-_{1b}(0,2), R^+_{2a}(3,2), R^-_{2a}(3,0), R^-_{3*}(0,0),\\
&R^+_{0*}(1,0), R^+_{1a}(1,2), R^-_{1a}(0,2), R^+_{2b}(3,2), R^-_{2b}(3,0), R^-_{3*}(0,0),\\
&R^+_{0*}(1,0), R^+_{1b}(1,2), R^-_{1b}(0,2), R^+_{2c}(3,2), R^-_{2c}(3,0), R^-_{3*}(0,0)).
\end{aligned}
$$

Cycle $\bar{C}^m_{d,L}$ has the following characteristics:

1. It begins with all machines occupied and the robot at the input buffer while holding no part.

2. The number of parts produced in one cycle is $\lambda = LCM[m_1, m_2, \ldots, m_m]$, the least common multiple of the number of machines in each stage.

3. For each stage i, the unloading of its machines is ordered alphabetically, beginning with machine M_{ia} in the first block.

4. Each machine is loaded immediately after it is unloaded. That is, state $R^-_{i\ell}(i+1,0)$ immediately follows state $R^+_{i\ell}(i+1,i)$, $i = 1, \ldots, m$; $\ell = a, \ldots, \alpha(m_i)$.

5. After loading a machine in stage i, the robot travels to stage $i+1$, $i = 1, \ldots, m$. If $1 \leq i \leq m-1$, then the robot unloads and loads some particular machine in stage $i+1$. If $i = m$, then the robot places the part into the output buffer (O), travels to the input buffer (I), and collects a new part which it carries to a machine in stage 1.

Note that characteristics 2, 3, and 4 are analogous to those of LCM cycles (Section 5.1.3). It follows that $\bar{C}^m_{d,L}$ is feasible and that for each stage i, each of its machines is used λ/m_i times.

In the following definition, $\phi(i)$ is the subscript of the next machine to be unloaded at stage i. If $\phi(i)$ is incremented to $\alpha(m_i+1)$, then set $\phi(i) = a$.

Cycle $\bar{C}^m_{d,L}$
Begin
    Set $\phi(i) = a$, $i = 1, \ldots, m$.
    For part $k = 1$ to $\lambda = LCM[m_1, m_2, \ldots, m_m]$ do:
    Begin
        ϵ: robot unloads a part from I.
        For $i = 1$ to m do:
        Begin
            δ: robot moves to $M_{i,\phi(i)}$.
            $w_{i,\phi(i)}$: robot waits for the part on $M_{i,\phi(i)}$ to complete.
            ϵ: robot unloads $M_{i,\phi(i)}$.
            θ: robot switches to the other gripper.
            ϵ: robot loads $M_{i,\phi(i)}$.
            increment $\phi(i)$ to next letter.
        End (Next i)
        δ: robot moves to O.
        ϵ: robot loads finished part onto O.
        δ: robot moves to I.
    End (Next k)
End

The cycle time for $\bar{C}^m_{d,L}$ can easily be calculated as

$$T(\bar{C}^m_{d,L}) = \lambda(m+2)\delta + 2\lambda(m+1)\epsilon + m\lambda\theta + W, \qquad (5.10)$$

$$\text{where } W = \sum_{i=1}^{m} \sum_{\ell=a}^{\alpha(m_i)} \sum_{r=1}^{\lambda/m_i} w^r_{i\ell}$$

is the robot's total waiting time during one iteration of cycle $\bar{C}^m_{d,L}$. Thus, if the robot never waits at any machine for it to complete processing, the per unit cycle time of $\bar{C}^m_{d,L}$ is the same as that of $\bar{C}^m_d$, and it equals the lower bound of Theorem 5.5. We now derive an expression for W.

Let $\tau = (m+2)\delta + 2(m+1)\epsilon + m\theta$. The same logic that led to equation (4.2) shows that the robot's waiting time at usage $M^r_{i\ell}$ during $\bar{C}^m_{d,L}$ is

$$w^r_{i\ell} = \max\left\{0, p_i - m_i\tau + 2\epsilon + \theta - \sum_{j \in G^r_{i\ell}} w_j\right\},$$

where $G^r_{i\ell}$ is the set of usages that are unloaded between the loading and the unloading of $M^r_{i\ell}$. This yields a system of λm inequalities

$$w^r_{i\ell} + \sum_{j \in G^r_{i\ell}} w_j \geq X_i, \ \forall i, \ell, r,$$

where $w^r_{i\ell} \geq 0, \ \forall i, \ell, r$, and $X_i = p_i - m_i\tau + 2\epsilon + \theta$.

For the cycle in Example 5.4, we have the following system of inequalities:

$$\begin{array}{cccccccccccccccccc}
w^3_{1a} & & & & & & & & & & & & & & & +w^1_{2b} & +w^2_{1b} & +w^1_{2c} & \geq X_1 \\
w^3_{1a} & +w^2_{2a} & & & & & & & & & & & & & +w^2_{1a} & +w^1_{2b} & +w^2_{1b} & +w^1_{2c} & \geq X_2 \\
w^3_{1a} & +w^2_{2a} & +w^3_{1b} & & & & & & & & & & & & & & & +w^1_{2c} & \geq X_1 \\
w^3_{1a} & +w^2_{2a} & +w^3_{1b} & +w^2_{2b} & & & & & & & & & & & & & +w^2_{1b} & +w^1_{2c} & \geq X_2 \\
 & w^2_{2a} & +w^3_{1b} & +w^2_{2b} & +w^1_{1a} & & & & & & & & & & & & & & \geq X_1 \\
w^3_{1a} & +w^2_{2a} & +w^3_{1b} & +w^2_{2b} & +w^1_{1a} & +w^2_{2c} & & & & & & & & & & & & & \geq X_2 \\
 & & & w^2_{2b} & +w^1_{1a} & +w^2_{2c} & +w^1_{1b} & & & & & & & & & & & & \geq X_1 \\
 & & w^3_{1b} & +w^2_{2b} & +w^1_{1a} & +w^2_{2c} & +w^1_{1b} & +w^1_{2a} & & & & & & & & & & & \geq X_2 \\
 & & & & & w^2_{2c} & +w^1_{1b} & +w^1_{2a} & +w^2_{1a} & & & & & & & & & & \geq X_1 \\
 & & & & w^1_{1a} & +w^2_{2c} & +w^1_{1b} & +w^1_{2a} & +w^2_{1a} & +w^1_{2b} & & & & & & & & & \geq X_2 \\
 & & & & & & & w^1_{2a} & +w^2_{1a} & +w^1_{2b} & +w^2_{1b} & & & & & & & & \geq X_1 \\
 & & & & & & w^1_{1b} & +w^1_{2a} & +w^2_{1a} & +w^1_{2b} & +w^2_{1b} & +w^1_{2c} & & & & & & & \geq X_2
\end{array}$$

with $w^r_{i\ell} \geq 0, \ \forall i, \ell, r$.

We now show that a minimal solution to this system of inequalities yields

$$W \ = \ \max\left\{0, \max_{1 \leq i \leq m}\left\{\frac{\lambda}{m_i} X_i\right\}\right\}.$$

Each machine $M_{i\ell}$ has one inequality for each of its λ/m_i usages. It is easy to see that each variable appears exactly once in the λ/m_i inequalities corresponding to $M_{i\ell}$, $i = 1, \ldots, m$; $\ell = a, \ldots, \alpha(m_i)$. This is because in the cycle $\bar{C}^m_{d,L}$, given any machine $M_{i\ell}$, every usage of every other machine is unloaded exactly once between the loading and

unloading of some usage $M_{i\ell}^r, r \in \{1, ..., \lambda/m_i\}$. By adding the λ/m_i inequalities of machine $M_{i\ell}$, we obtain the following m inequalities:

$$W = \sum_{i=1}^{m} \sum_{\ell=a}^{\alpha(m_i)} \sum_{r=1}^{\lambda/m_i} w_{i\ell}^r \geq \frac{\lambda}{m_i} X_i, \quad i = 1, ..., m.$$

Since $W \geq 0$, we have $W \geq \max\{0, \max_{1\leq i\leq m}\{\lambda X_i/m_i\}\}$. We exhibit a feasible solution whose value is $\max\{0, \max_{1\leq i\leq m}\{\lambda X_i/m_i\}\}$. Let i' be such that $\lambda X_{i'}/m_{i'} = \max_{1\leq i\leq m}\{\lambda X_i/m_i\}$. Let the waiting time at each usage of each machine in stage i' be $X_{i'}/m_{i'}$. That is, $w_{i'\ell}^r = X_{i'}/m_{i'}, \forall\ \ell, r$, and let $w_{i\ell}^r = 0, i \neq i'$. In each of the λ inequalities corresponding to usages of machines in stage i', i.e., inequalities whose right-hand sides are $X_{i'}$, the left-hand side evaluates to $m_{i'}(X_{i'}/m_{i'}) = X_{i'}$. In all other inequalities, the left-hand side evaluates to $m_i X_{i'}/m_{i'} \geq m_i(X_i/m_i) = X_i$. Therefore, this is a feasible solution with $W = \max\{0, \max_{1\leq i\leq m}\{\lambda X_i/m_i\}\}$. Thus,

$$\begin{aligned}
W &= \sum_{i=1}^{m} \sum_{\ell=a}^{\alpha(m_i)} \sum_{r=1}^{\lambda/m_i} w_{i\ell}^r \\
&= \max\left\{0, \max_{1\leq i\leq m}\left\{\frac{\lambda}{m_i} X_i\right\}\right\} \\
&= \max\left\{0, \max_{1\leq i\leq m}\left\{\frac{\lambda}{m_i}(p_i - m_i\tau + 2\epsilon + \theta)\right\}\right\} \\
&= \max\left\{0, \max_{1\leq i\leq m}\left\{\frac{\lambda}{m_i} p_i - \lambda(m+2)\delta - 2\lambda\left(m + 1 - \frac{1}{m_i}\right)\epsilon \right.\right. \\
&\qquad \left.\left. -\lambda\left(m - \frac{1}{m_i}\right)\theta\right\}\right\}.
\end{aligned}$$

Combining this result with equation (5.10), we have

$$\begin{aligned}
T(\bar{C}_{d,L}^m) = \max\{&\lambda(m+2)\delta + 2\lambda(m+1)\epsilon + m\lambda\theta, \\
&\max_{1\leq i\leq m}\left\{\frac{\lambda}{m_i}(p_i + 2\epsilon + \theta)\right\}\Big\}.
\end{aligned}$$

We have thus proved the following result.

THEOREM 5.6 *In a robotic cell with parallel machines and a dual-gripper robot, $\bar{C}_{d,L}^m$ is optimal among all k-unit cyclic schedules ($k \geq 1$) under the assumption that $\theta \leq \delta$ and $p_i \geq \delta$, $i = 1, \ldots, m$.*

REMARK 5.1 Because the lower bound for the per unit cycle time in a robotic cell with parallel machines and a single-gripper robot, according to Theorem 5.1, is

$$\frac{T(\pi)}{k} \geq \max\left\{2(m+1)\epsilon + \sum_{i=1}^{m}\min\{p_i,\delta\} + (m+2)\delta, \max_{1\leq i\leq m}\left\{\frac{p_i+3\delta+4\epsilon}{m_i}\right\}\right\},$$

$\bar{C}^m_{d,L}$ achieves a greater throughput than all the single-gripper k-unit cycles in cells with parallel machines that satisfy $\theta \leq \min\{\delta, p_i\}$, $i = 1,\ldots,m$.

REMARK 5.2 We know of no rigorous studies of robotic cells with parallel machines and additive travel times. Hence, the structure of an optimal cycle or a lower bound on the per unit cycle time are not known for such cells, either for single-gripper or dual-gripper robots.

5.2.3 Improvement from Using a Dual-Gripper Robot or Parallel Machines

We first note that starting from the single-gripper simple robotic cell model, there are essentially three different ways in which managers can attempt to improve productivity (keeping other process-related procedures fixed):

1. by considering the use of a dual-gripper robot,
2. by considering the installation of parallel machines at bottleneck stages, and
3. by considering the use of both a dual-gripper robot and parallel machines.

To assist production managers with the decision making, we provide an analysis by dividing the discussion into three subsections that reflect the three options listed above. Our analysis attempts to obtain a bound on the potential decrease in the per unit cycle time (or, equivalently, an increase in the throughput) if a particular option is chosen. For a given cell, the computations required to do the analysis are straightforward, and the resulting bounds will enable managers to compare these

options. Throughout our analysis we assume that $\theta \leq \delta$ and $p_i \geq \delta$, as these conditions hold in most manufacturing applications. Following our analysis, we illustrate it on two cells we encountered in our work with a Dallas-area semiconductor equipment manufacturer.

5.2.3.1 Installing a Dual-Gripper Robot in a Simple Robotic Cell

In a simple robotic cell, we compare the two 1-unit cycles π_D (Chapter 3) and $\bar{C}_d^m$ (Chapter 4). Recall that $T(\pi_D) = \max\{2(m+1)(\delta + \epsilon), \max_i\{p_i\} + 3\delta + 4\epsilon\}$, and, if $p_i \geq \delta, \forall i$, then it is optimal in a cell with a single-gripper robot.

Before we provide the details of the analysis, we summarize the results:

- If $T(\pi_D) = 2(m+1)(\delta+\epsilon)$, then the cell is robot-constrained and the per unit cycle time can be decreased by up to $m(\delta - \theta)$ by using a dual-gripper robot and implementing cycle $\bar{C}_d^m$.

- If $T(\pi_D) = \max_i\{p_i\} + 3\delta + 4\epsilon$, then the decrease in the cycle time from using a dual-gripper robot is at best $3\delta + 2\epsilon - \theta$.

We now provide the analysis on a case-by-case basis. The cases concern the relations between the arguments in the per unit cycle time expressions for π_D and $\bar{C}_d^m$:

$$\begin{aligned} T(\pi_D) &= \max\{2(m+1)(\delta+\epsilon), \max_{1\leq i\leq m}\{p_i\} + 3\delta + 4\epsilon\}, \\ T(\bar{C}_d^m) &= \max\{2(m+1)\epsilon + (m+2)\delta + m\theta, \max_{1\leq i\leq m}\{p_i\} + 2\epsilon + \theta\}. \end{aligned}$$

Recall from Corollary 4.6 that cycle $\bar{C}_d^m$ is optimal for a simple robotic cell with a dual-gripper robot among all k-unit cyclic schedules ($k \geq 1$) under the assumption that $\theta \leq \delta$ and $p_i \geq \delta$, $i = 1, \ldots, m$.

Case 1.

$$\begin{aligned} 2(m+1)(\delta+\epsilon) &\geq \max\{p_i\} + 3\delta + 4\epsilon, \\ 2(m+1)\epsilon + (m+2)\delta + m\theta &\geq \max\{p_i\} + 2\epsilon + \theta. \end{aligned}$$

Using a dual-gripper robot will decrease the per unit cycle time by $m(\delta - \theta)$.

Case 2.

$$\begin{aligned} 2(m+1)(\delta+\epsilon) &\geq \max\{p_i\} + 3\delta + 4\epsilon \\ &\geq \max\{p_i\} + 2\epsilon + \theta \\ &\geq 2(m+1)\epsilon + (m+2)\delta + m\theta. \end{aligned}$$

Using a dual-gripper robot will decrease the per unit cycle time by $x = 2(m+1)(\delta+\epsilon) - [\max\{p_i\} + 2\epsilon + \theta]$. Note that $3\delta + 2\epsilon - \theta \leq x \leq m(\delta - \theta)$.

Case 3.

$$\begin{aligned} \max\{p_i\} + 3\delta + 4\epsilon &\geq 2(m+1)(\delta+\epsilon) \geq \\ 2(m+1)\epsilon &+ (m+2)\delta + m\theta \geq \max\{p_i\} + 2\epsilon + \theta. \end{aligned}$$

The improvement in the cycle time from a dual-gripper robot is $x = \max\{p_i\} + 3\delta + 4\epsilon - [2(m+1)\epsilon + (m+2)\delta + m\theta]$. Note that $m(\delta - \theta) \leq x \leq 3\delta + 2\epsilon - \theta$.

Case 4.

$$\begin{aligned} \max\{p_i\} + 3\delta + 4\epsilon &\geq 2(m+1)(\delta+\epsilon) \\ \max\{p_i\} + 2\epsilon + \theta &\geq 2(m+1)\epsilon + (m+2)\delta + m\theta. \end{aligned}$$

The improvement in the cycle time in this case is $3\delta + 2\epsilon - \theta$.

Table 5.1 lists eight different simple robotic cells and their data. These were chosen because they cover the four different cases just discussed. Additionally, based on our experience with a robotic cell manufacturer, these are reasonable examples. For each cell, the table lists $T(\pi_D)$ for a single-gripper robot and $T(\bar{C}_d^m)$ for a dual-gripper robot. The table also shows the case to which each cell belongs to as well as the percentage improvement in throughput from using a dual-gripper robot.

5.2.3.2 Installing Parallel Machines in a Single-Gripper Robot Cell

Recall that in the case which we consider (i.e., $p_i \geq \delta, \forall i$), π_{LD} is optimal in a single-gripper robotic cell. If $\max\{p_i\} + 3\delta + 4\epsilon > 2(m+1)(\delta+\epsilon)$, then this cell is process-constrained at stage i^*, where $p_{i^*} = \max\{p_i\}$. Its throughput can be increased by adding parallel machines

m	δ	ϵ	θ	$\max\{p_i\}$	$T(\pi_D)$	$T(\bar{C}_d^m)$	% improvement	
5	3	1	1	20	48	38	20.83	Case 1
15	3	1	1	50	128	98	23.44	Case 1
15	10	1	1	150	352	217	38.35	Case 1
10	3	1	1	70	88	73	17.05	Case 2
5	3	1	2	38	51	43	15.69	Case 3
5	3	1	1	50	63	53	15.87	Case 4
5	3	1	3	50	63	55	12.70	Case 4
15	3	1	1	150	163	153	6.13	Case 4

Table 5.1. Cycle-Time Improvement from using a Dual-Gripper Robot in a Simple Robotic Cell.

to stage i^*. The best possible per unit cycle time is $2(m+1)(\delta+\epsilon)$. This is achieved if, for each stage i, the number of parallel machines m_i satisfies (Section 5.1.4.2) $(p_i+3\delta+4\epsilon)/m_i \leq 2(m+1)(\delta+\epsilon)$. Therefore,

$$m_i = \left\lceil \frac{p_i + 3\delta + 4\epsilon}{2(m+1)(\delta+\epsilon)} \right\rceil$$

is the minimum number of machines needed at stage i, $\forall i$. If $\max\{p_i\} + 3\delta+4\epsilon \leq 2(m+1)(\delta+\epsilon)$, then there will be no benefit from using parallel machines in this cell.

Table 5.2 lists the same eight cells as those in Table 5.1. For each, it gives $T(\pi_D)$ for a simple robotic cell, the best possible value for $T(\pi_{LD})$ in a robotic cell with parallel machines, and the percentage increase in throughput realized by using parallel machines. Note that only those single-gripper robotic cells in Cases 3 or 4 (i.e., process-constrained) benefit from the addition of parallel machines.

5.2.3.3 Installing a Dual-Gripper Robot in a Single-Gripper Robotic Cell with Parallel Machines

We have seen that dual-gripper robots improve the throughput of robot-constrained cells, and that parallel machines improve the throughput of process-constrained cells. There are also instances in which the use of both can lead to a greater improvement in throughput. Suppose we have added parallel machines to a process-constrained single-gripper robotic cell so that $T(\pi_{LD})/\lambda = 2(m+1)(\delta+\epsilon)$. Throughput can now be increased by using a dual-gripper robot and cycle $\bar{C}_{d,L}^m$. Having added a

m	δ	ϵ	θ	$\max\{p_i\}$	$T(\pi_D)$	$T(\pi_{LD})$	% improvement	
5	3	1	1	20	48	48	0	Case 1
15	3	1	1	50	128	128	0	Case 1
15	10	1	1	150	352	352	0	Case 1
10	3	1	1	70	88	88	0	Case 2
5	3	1	2	38	51	48	5.88	Case 3
5	3	1	1	50	63	48	23.81	Case 4
5	3	1	3	50	63	48	23.81	Case 4
15	3	1	1	150	163	128	21.47	Case 4

Table 5.2. Cycle-Time Improvement from Adding Parallel Machines to a Simple Robotic Cell with a Single-Gripper Robot.

dual-gripper robot, if

$$\frac{p_{i^*}+2\epsilon+\theta}{m_{i^*}} = \max_{1\le i\le m}\frac{p_i+2\epsilon+\theta}{m_i} \ge 2(m+1)\epsilon+(m+2)\delta+m\theta,$$

then additional parallel machines at stage i^* will increase throughput further. The minimum possible per unit cycle time $T(\bar{C}^m_{d,L})/\lambda = 2(m+1)\epsilon+(m+2)\delta+m\theta$ is achieved with the minimum number of machines if

$$m_i = \left\lceil \frac{p_i+2\epsilon+\theta}{2(m+1)\epsilon+(m+2)\delta+m\theta} \right\rceil, \ \forall i. \tag{5.11}$$

We now analyze the benefits of using a dual-gripper robot and parallel machines on a case-by-case basis. We show that for all cases these enhancements will reduce per unit cycle time by at least $m(\delta-\theta)$. The cases concern the relations between the arguments in the per unit cycle time expressions for π_{LD} and $\bar{C}^m_{d,L}$:

$$\begin{aligned}
\tfrac{T(\pi_{LD})}{\lambda} &= \max\left\{2(m+1)(\delta+\epsilon), \max_{1\le i\le m}\left\{\frac{p_i+3\delta+4\epsilon}{m_i}\right\}\right\},\\
\tfrac{T(\bar{C}^m_{d,L})}{\lambda} &= \max\left\{2(m+1)\epsilon+(m+2)\delta+m\theta, \max_{1\le i\le m}\left\{\frac{p_i+2\epsilon+\theta}{m_i}\right\}\right\}.
\end{aligned}$$

Case 1.

$$\begin{aligned}
2(m+1)(\delta+\epsilon) &\ge \max_{1\le i\le m}\frac{p_i+3\delta+4\epsilon}{m_i},\\
2(m+1)\epsilon+(m+2)\delta+m\theta &\ge \max_{1\le i\le m}\frac{p_i+2\epsilon+\theta}{m_i}.
\end{aligned}$$

We have seen in the previous two sections that using additional parallel machines will have no benefit, and using a dual-gripper robot will decrease the per unit cycle time by $m(\delta - \theta)$.

Case 2.

$$\begin{aligned} 2(m+1)(\delta+\epsilon) &\geq \max_{1\leq i\leq m} \frac{p_i + 3\delta + 4\epsilon}{m_i} \\ &\geq \max_{1\leq i\leq m} \frac{p_i + 2\epsilon + \theta}{m_i} \\ &\geq 2(m+1)\epsilon + (m+2)\delta + m\theta. \end{aligned}$$

Using additional parallel machines will provide no benefit in a single-gripper robotic cell, but it will in a dual-gripper robotic cell. Using a dual-gripper robot will decrease the per unit cycle time by

$$x = 2(m+1)(\delta+\epsilon) - \max_{1\leq i\leq m} \frac{p_i + 2\epsilon + \theta}{m_i}.$$

Note that $(3\delta + 2\epsilon - \theta)/m_{i^*} \leq x \leq m(\delta - \theta)$. Using a dual-gripper robot and additional parallel machines will decrease the per unit cycle time by $m(\delta - \theta)$.

Case 3.

$$\begin{aligned} \max_{1\leq i\leq m} \frac{p_i + 3\delta + 4\epsilon}{m_i} &\geq 2(m+1)(\delta+\epsilon) \\ \geq 2(m+1)\epsilon &+ (m+2)\delta + m\theta \geq \max_{1\leq i\leq m} \frac{p_i + 2\epsilon + \theta}{m_i}. \end{aligned}$$

Adding parallel machines will help in the single-gripper robot case, but not in the dual-gripper robot case. The benefit of implementing dual-gripper robot is

$$x = \max_{1\leq i\leq m} \frac{p_i + 3\delta + 4\epsilon}{m_i} - [2(m+1)\epsilon + (m+2)\delta + m\theta].$$

Note that $m(\delta - \theta) \leq x \leq (3\delta + 2\epsilon - \theta)/m_{i^*}$.

Case 4.

$$\begin{aligned} \max_{1\leq i\leq m} \frac{p_i + 3\delta + 4\epsilon}{m_i} &\geq 2(m+1)(\delta+\epsilon), \\ \max_{1\leq i\leq m} \frac{p_i + 2\epsilon + \theta}{m_i} &\geq 2(m+1)\epsilon + (m+2)\delta + m\theta. \end{aligned}$$

The benefit of using a dual-gripper robot is $(3\delta + 2\epsilon - \theta)/m_{i^*}$. The benefit of using a dual-gripper robot and parallel machines is

$$\begin{aligned} & \frac{p_{i^*} + 3\delta + 4\epsilon}{m_{i^*}} - [2(m+1)\epsilon + (m+2)\delta + m\theta] \\ \geq \quad & \max\left\{\frac{3\delta + 2\epsilon - \theta}{m_{i^*}}, m(\delta - \theta)\right\}. \end{aligned}$$

For each of the eight cells previously analyzed, Table 5.3 lists $T(\pi_D)$ for a simple robotic cell with a single-gripper robot, $T(\bar{C}^m_{d,L})$ for a robotic cell with parallel machines (where m_i is specified by equation (5.11), $\forall i$) and a dual-gripper robot, the percentage throughput improvement realized by using both parallel machines and a dual-gripper robot, and to which case each belongs.

m	δ	ϵ	θ	max p_i	$T(\pi_D)$	$T(\bar{C}^m_{d,L})$	% improvement	
5	3	1	1	20	48	38	20.83	Case 1
15	3	1	1	50	128	98	23.44	Case 1
15	10	1	1	150	352	217	38.35	Case 1
10	3	1	1	70	88	68	22.73	Case 2
5	3	1	2	38	51	43	15.69	Case 3
5	3	1	1	50	63	38	39.68	Case 4
5	3	1	3	50	63	48	23.81	Case 4
15	3	1	1	150	163	98	39.88	Case 4

Table 5.3. Cycle-Time Improvement from Adding Parallel Machines and a Dual-Gripper Robot to a Simple Robotic Cell with a Single-Gripper Robot.

We have previously seen that in a simple robotic cell, using a dual-gripper robot and cycle $\bar{C}^m_d$ increases throughput. In addition, in a robotic cell with parallel machines, using a dual-gripper robot and cycle $\bar{C}^m_{d,L}$ increases throughput. We can now state that adding parallel machines to appropriate stages in a single-gripper robotic cell increases throughput for Cases 3 and 4, and that adding parallel machines to appropriate stages in a dual-gripper robotic cell increases throughput for Cases 2 and 4.

5.2.3.4 An Illustration on Data from Implemented Cells

We now consider two simple robotic cells with single-gripper robots that have been designed and developed by a semiconductor equipment manufacturer. We illustrate our analysis on these cells by assessing the gains from using a dual-gripper robot and/or parallel machines.

Cell Data. The first cell has 11 processing stages. The vector of processing times (in seconds) for these stages is $p = (50.00, 52.24, 59.00, 70.13, 52.24, 25.00, 70.13, 52.24, 86.00, 70.13, 52.24)$. The second cell has 10 processing stages. The vector of processing times (in seconds) for these stages is $p = (37.5, 70.13, 37.24, 39.5, 100.13, 42.24, 13, 100.13, 42.24, 65)$. These cells are used in photolithography to transfer electronic circuit patterns onto silicon wafers. The number of steps varies to accommodate differing requirements, e.g., top anti-reflective coating, bottom anti-reflective coating, tri-level, etc. Because of confidentiality agreements, we cannot be more specific about each cell's usage.

For both of these cells, the inter-machine travel times, although not a single value, are roughly equal. For each, the maximum, average, and minimum inter-machine travel times are 4.15 seconds, 3.9 seconds, and 3.65 seconds, respectively. Thus, the relative difference between the maximum (δ^u) and minimum (δ^l) inter-machine travel time is $(\delta^u - \delta^l)/\delta^u = 12.0\%$. The constant travel-time robotic cell model is therefore appropriate for both of these cells. We are unable to provide the complete matrix of inter-machine travel times because of a data confidentiality agreement with the company. For our purposes, we let $\delta = 3.9$ be the inter-machine travel time for each cell. For both cells, the load and unload times are $\epsilon = 0.5$ seconds, and the dual-gripper switch time is $\theta = 0.5$ seconds.

Results. We first measure the throughput improvements for these two cells using the constant travel-time model. We then derive theoretical lower bounds for these improvements when the actual travel times are used.

The first cell belongs to Case 2. With no parallel machines and a single-gripper robot, the optimal per unit cycle time for this cell is $T(\pi_D) = 105.6$ seconds. Adding parallel machines while keeping the single-gripper robot will not increase throughput. However, when using

a dual-gripper robot, the optimal cycle time is $T(\bar{C}_d^m) = 87.5$ seconds, a reduction of 17.1%. If we now add one parallel machine each to stages 4, 7, 9, and 10, the optimal per unit cycle time with a dual-gripper robot is $T(\bar{C}_{d,L}^m) = 68.2$ seconds, which represents a reduction of 35.4% over that of the simple robotic cell with a single-gripper robot.

The second cell belongs to Case 4. With no parallel machines and a single-gripper robot, the optimal per unit cycle time for this cell is $T(\pi_D) = 113.83$ seconds. Adding one parallel machine each to stages 5 and 8 while keeping the single-gripper robot will reduce the per unit cycle time to 96.80 seconds, a decrease of 15.0%. Using a dual-gripper robot with no parallel machines will reduce the cycle time to 101.63 seconds, a decrease of 10.7%. If we use a dual-gripper robot and add one parallel machine each to stages 2, 5, 8, and 10, the per unit cycle time is $T(\bar{C}_{d,L}^m) = 62.8$, a reduction of 44.8% over the simple robotic cell with a single-gripper robot. The results for both of these cells are summarized in Table 5.4.

machines	simple robotic cell			parallel machines		parallel machines	
grippers	single	dual-grippers		single-gripper		dual-grippers	
metric	$T(\pi_D)$	$T(\bar{C}_d^m)$	imprv.	$T(\pi_{LD})$	imprv.	$T(\bar{C}_{d,L}^m)$	imprv.
Cell 1	105.6	87.5	17.1%	105.6	0.00%	68.2	35.4%
Cell 2	113.8	101.6	10.7%	96.8	15.0%	62.8	44.8%

Table 5.4. Cycle-Time Improvements on Two Cells Used in Semiconductor Wafer Fabrication.

We now establish theoretical lower bounds on the throughput improvements with the actual travel times. To compute the worst possible improvement realized from implementing dual-gripper robots in these simple robotic cells, we compare the difference of $T(\pi_D)$ computed with δ^l and $T(\bar{C}_d^m)$ computed with δ^u, divided by $T(\pi_D)$ computed with δ^u. For the first cell, the result is

$$\frac{T(\pi_D(\delta^l)) - T(\bar{C}_d^m(\delta^u))}{T(\pi_D(\delta^u))} = \frac{99.6 - 87.5}{111.6} = 10.8\%.$$

For the second cell the value is 10.1%. Similarly, the lower bounds on the throughput improvements from adding dual-gripper robots and parallel

machines to the simple robotic cells with single-gripper robots are 31.0% and 43.4%.

A rough calculation of the financial benefits of these improvements can be made as follows. A typical target throughput for a simple robotic cell with a single-gripper robot is around 110 wafers per hour. The cost of a wafer is typically in the range \$10,000–\$30,000. Therefore, a conservative estimate of the improvement in revenues is

$$100\ \frac{\text{wafers}}{\text{hour}} \times 10.7\% \times \frac{\$10,000}{\text{wafer}} \times 32\ \frac{\text{hours}}{\text{week}} = \$3,424,000/\text{week}.$$

Given that the marginal cost of adding a dual-gripper to a single-gripper robot is approximately \$10,000, and the required software changes would require labor on the order of \$200,000 [133], the time required to recapture the capital investment is very small. A similar calculation can be done for the addition of parallel machines.

Chapter 6

MULTIPLE-PART-TYPE PRODUCTION: SINGLE-GRIPPER ROBOTS

We now examine single-gripper robotic cells that process lots containing different types of parts. In general, the processing times at the machines differ for the different types of parts. Such implementations are more commonly used by medium-sized discrete part manufacturers. Multiple parts are processed in a single lot in order to have enough volume to use the cell efficiently (Ramanan [133]). All the results in this chapter are for additive travel-time cells with a linear or semicircular layout.

In accordance with just-in-time manufacturing, it is typical for the relative proportions of the part-types in each lot to be approximately the same as the relative proportions of their demand. Consequently, research has focused on cycles which contain a minimal part set (MPS) that has these same proportions. An MPS is a minimum cardinality set of parts such that the relative proportions of the parts are the same as those of their demands during a planning horizon. For example, if the demand for a company's three products is divided so that product A has 40%, product B has 35%, and product C has 25%, then the MPS has 20 parts: 8 of product A, 7 of product B, and 5 of product C. In practice, the size of an MPS can exceed 50 parts (Wittrock [158]).

We will assume that the cell processes k different part-types and that one MPS produces r_i parts of type i, $i = 1, \ldots, k$. The total number of finished parts in one MPS cycle is $n = r_1 + \cdots + r_k$; these are collectively referred to as *MPS parts*. The schedule of production, referred to as the *MPS part schedule* (or simply a *part schedule*), is specified by a

permutation σ, with $P_{\sigma(i)}$ being the part scheduled in the ith position of $\sigma, i = 1, \ldots, n$. Our objective in this chapter is to minimize the average (cycle) time T required to produce one MPS in a cyclic production environment. As before, the throughput rate μ is defined as n/T.

6.1 MPS Cycles and CRM Sequences

We define a general *MPS cycle* as a sequence of robot moves during which the MPS parts enter the system from the Input, are processed at machines M_1, ..., M_m, and leave the system at the Output, after which the system returns to its initial state. A specific MPS cycle can be defined by the schedule σ in which the MPS parts enter the cell from the Input and the sequence of robot actions within the cell. An *MPS robot move sequence* (or simply a robot move sequence) is a sequence of robot actions performed during an MPS cycle. Thus, given an MPS robot move sequence and an MPS part schedule, the corresponding MPS cycle can be constructed. Single-part-type production is the special case when the MPS consists of a single part.

Concatenated Robot Move Sequences (CRM sequences) form a class of MPS cycles in which the robot move sequence is the same 1-unit cycle of robot actions repeated n times, where n is the total number of parts to be produced in an MPS cycle (Sriskandarajah et al. [146]). For example, for $m = 3$ and $n = 3$, the CRM sequence corresponding to the sequence $\pi_4 = (A_0, A_3, A_1, A_2)$ is

$$(\pi_4, \pi_4, \pi_4) = (A_0, A_3, A_1, A_2, A_0, A_3, A_1, A_2, A_0, A_3, A_1, A_2).$$

Let $CRM(\pi_j)$ denote the CRM sequence associated with the 1-unit sequence π_j. When no confusion arises in doing so, we simply use π_j to denote $CRM(\pi_j)$. Appendix A provides a list of all 1-unit cycles for $m = 2, 3$, and 4. Ideally, the number of parts to be produced, n, should also be specified in the notation for a CRM sequence. However, all the results in this chapter are for arbitrary values of n; we therefore omit the specification.

In Section 6.2, we show using an example that the MPS cycle based on the CRM sequence formed by the best 1-unit robot move sequence can be suboptimal in a two-machine cell producing multiple part-types. We therefore provide an efficient algorithm for finding the robot move sequence and the part schedule which jointly minimize the production

cycle time. In Section 6.3, a number of realistic and easily solvable special cases are identified for part scheduling problems in three-machine cells. Since all of our results are derived for a robotic cell operating in a steady state, we also study ways in which a robotic cell converges to such a state in Section 6.4. For two of the six CRM sequences in three-machine cells, the recognition version of the part scheduling problem is strongly NP-complete, as is the general part scheduling problem that is not restricted to any robot move sequence. We prove these results in Section 6.5. Section 6.6 generalizes the complexity results for cells with $m \geq 4$. In particular, this section determines the tractability of the part scheduling problems associated with various CRM sequences in an m-machine cell. Of the $m!$ cycles of this type in an m-machine cell, exactly $2m-2$ have an associated part scheduling problem that is efficiently solvable; the part scheduling problems associated with the other cycles are intractable. Among the intractable part scheduling problems, we identify those that can be easily (in the sense defined later in Section 6.6.3) formulated as Traveling Salesman Problems (TSPs). Sections 6.7–6.8 consider several scheduling problems that have been shown to be intractable in the previous sections and provide simple but computationally effective heuristics to solve industry-sized instances. In particular, Sections 6.7.1 and 6.7.2 describe heuristics for intractable part scheduling problems in three-machine cells. The general three- and four-machine problems are examined in Sections 6.7.4 and 6.8, respectively. Heuristics for larger cells are discussed in Section 6.8. Section 6.9 considers the design of robotic cells within a larger manufacturing system in which several machines need to be arranged into cells that are separated by intermediate buffers.

We recall the notation to be used throughout this chapter:

a_k, b_k, c_k, d_k: the processing times of part P_k on machines M_1, M_2, M_3, M_4, respectively, when $m \leq 4$. For cells with five or more machines, the processing time of part P_k on machine M_j is denoted by $p_{k,j}$.

δ_i: the time taken by the robot to travel between two consecutive machines M_{i-1} and M_i, $1 \leq i \leq m+1$.

η: the time saved by not stopping at an intermediate machine during a robot travel between two nonconsecutive machines.

ϵ_1: the time taken by the robot to pick up a part at I.

ϵ_{2i}: the time taken by the robot to load a part onto machine M_i.

ϵ_{2i+1}: the time taken by the robot to unload a part from machine M_i.

ϵ_{2m+2}: the time taken by the robot to drop a part at O.

σ: the order in which parts of an MPS are processed repetitively.

$P_{\sigma(j)}$: the jth part to be produced in the schedule σ.

$a_{\sigma(j)}$ (resp., $b_{\sigma(j)}, c_{\sigma(j)}, d_{\sigma(j)}$): the processing time of the jth part in schedule σ on machine M_1 (resp., M_2, M_3, M_4).

$(\chi_1, \ldots, \chi_m, M_h^j)$: the current state of the system, where $\chi_i = \phi$ (resp., Ω) if machine M_i is free (resp., occupied by a part). The robot has just loaded (resp., unloaded) machine M_h if $j = -$ (resp., $+$).

6.2 Scheduling Multiple Part-Types in Two-Machine Cells

For additive travel-time cells (problem $RF_2|(free,A,MP,cyclic\text{-}n)|\mu$), Hall et al. [75] address the two problems − part scheduling and robot move sequencing − simultaneously. They first show that, in general, CRM sequences are not optimal robot move sequences. Rather, it is often better to selectively switch between π_1 and π_2, where π_1 and π_2 are the two robot move sequences for 1-unit cycles in two-machine cells (see Chapter 3 and Appendix A).

We begin our analysis of the two-machine robotic cell problem, $RF_2|(free,A,\ MP,cyclic\text{-}n)|\mu$, by showing that the optimal solution is not generally given by an MPS cycle using a CRM sequence. The cycle time minimization problem for multiple part-types then becomes one of deciding under which robot move sequence (π_1 or π_2 or a combination of π_1 and π_2) to process each part and specifying how to switch from π_1 to π_2 or vice versa, while simultaneously choosing the optimal part schedule. An $O(n^4)$ algorithm that solves this problem optimally is provided.

Let $T^h_{j\sigma(i)\sigma(i+1)}$ be the time between the loading of part $P_{\sigma(i)}$ on machine M_h, and the loading of part $P_{\sigma(i+1)}$ on M_h, using the CRM sequence π_j, where $j \in \{1,2\}$ and $h \in \{1,2\}$. We use w^i_j to denote the

waiting time of the robot before unloading part $P_{\sigma(i)}$ at machine M_j. We now derive expressions for $T^h_{1\sigma(i)\sigma(i+1)}$ and $T^h_{2\sigma(i)\sigma(i+1)}$.

Starting from the initial state $E = (\emptyset, \Omega, M_2^-)$ (see Chapter 2), where the robot has just completed loading part $P_{\sigma(i)}$ onto M_2, the robot move sequence π_1 includes the following activities: wait for $P_{\sigma(i)}$ at M_2 ($b_{\sigma(i)}$), unload (ϵ_5), move to O (δ_3), drop $P_{\sigma(i)}$ (ϵ_6), move to I ($\delta_1+\delta_2+\delta_3-2\eta$), pick up $P_{\sigma(i+1)}$ (ϵ_1), move to M_1 (δ_1), load (ϵ_2), wait for $P_{\sigma(i+1)}$ ($a_{\sigma(i+1)}$), unload (ϵ_3), move to M_2 (δ_2), and load (ϵ_4). Thus, we have

$$\begin{aligned} T^2_{1\sigma(i)\sigma(i+1)} &= b_{\sigma(i)} + \epsilon_5 + \delta_3 + \epsilon_6 + (\delta_1 + \delta_2 + \delta_3 - 2\eta) \\ &\quad + \epsilon_1 + \delta_1 + \epsilon_2 + a_{\sigma(i+1)} + \epsilon_3 + (\delta_2) + \epsilon_4. \end{aligned}$$

Starting from the initial state $E = (\emptyset, \Omega, M_2^-)$, where the robot has just completed loading part $P_{\sigma(i)}$ onto M_2, the robot move sequence π_2 includes the following activities: move to I ($\delta_1 + \delta_2 - \eta$), pick up part $P_{\sigma(i+1)}$ (ϵ_1), move to M_1 (δ_1), load (ϵ_2), move to M_2 (δ_2), wait (if necessary) for $P_{\sigma(i)}$ (w_2^i), unload (ϵ_5), move to O (δ_3), drop $P_{\sigma(i)}$ (ϵ_6), move to M_1 ($\delta_2 + \delta_3 - \eta$), wait (if necessary) for $P_{\sigma(i+1)}$ (w_1^{i+1}), unload (ϵ_3), move to M_2 (δ_2), and load (ϵ_4). Therefore,

$$\begin{aligned} T^2_{2\sigma(i)\sigma(i+1)} &= (\delta_1 + \delta_2 - \eta) + \epsilon_1 + \delta_1 + \epsilon_2 + (\delta_2) + w_2^i + \epsilon_5 + \delta_3 + \epsilon_6 \\ &\quad + (\delta_2 + \delta_3 - \eta) + w_1^{i+1} + \epsilon_3 + (\delta_2) + \epsilon_4, \text{ where} \end{aligned}$$

$$\begin{aligned} w_1^{i+1} &= \max\{0, a_{\sigma(i+1)} - w_2^i - \epsilon_5 - \epsilon_6 - 2\delta_2 - 2\delta_3 + \eta\}, \\ w_2^i &= \max\{0, b_{\sigma(i)} - \epsilon_1 - \epsilon_2 - 2\delta_1 - 2\delta_2 + \eta\}. \end{aligned}$$

By combining the expressions for w_1^{i+1} and w_2^i, we obtain the following result.

LEMMA 6.1 *The cycle time expressions for RF$_2$|(free,A,MP,CRM)|μ are given by*

$$\begin{aligned} T^2_{1\sigma(i)\sigma(i+1)} &= \rho + b'_{\sigma(i)} + a'_{\sigma(i+1)}, \\ T^2_{2\sigma(i)\sigma(i+1)} &= \rho + \max\{\nu, b'_{\sigma(i)}, a'_{\sigma(i+1)}\}, \text{ where} \\ a'_i &= a_i + \epsilon_1 + \epsilon_2 + 2\delta_1 - \eta, \\ b'_i &= b_i + \epsilon_5 + \epsilon_6 + 2\delta_3 - \eta, i = 1, \ldots, n, \\ \rho &= \epsilon_3 + \epsilon_4 + 2\delta_2, \\ \nu &= 2\sum_{i=1}^{3} \delta_i + \epsilon_1 + \epsilon_2 + \epsilon_5 + \epsilon_6 - 2\eta. \end{aligned}$$

Proof. Follows from the above analysis. ∎

In both sequences π_1 and π_2, a pair of parts $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ is involved. The reader should also note that the state $E = (\emptyset, \Omega, M_2^-)$, in which the robot has just finished loading a part on M_2, is the only state common to π_1 and π_2. Hence, this is the only state in which switching between π_1 and π_2 can be achieved without wasteful robot moves. In $E = (\emptyset, \Omega, M_2^-)$, the robot has two choices:

a. Wait and unload M_2 (as in sequence π_1).

b. Move to I (as in sequence π_2).

Because of the possibility of switching between sequences at $E = (\emptyset, \Omega, M_2^-)$, a part $P_{\sigma(i)}$ may be processed using both sequences. The following possibilities exist:

1. During the processing of $P_{\sigma(i)}$, the robot uses sequence π_i, $i \in \{1, 2\}$, on both machine M_1 and machine M_2. In this case, we say that $P_{\sigma(i)}$ is processed using sequence π_i.

2. The robot uses sequence π_1 during the processing of $P_{\sigma(i)}$ on M_1 and sequence π_2 during the processing of $P_{\sigma(i)}$ on M_2. We say that $P_{\sigma(i)}$ is processed using sequence π_{1-2}.

3. The robot uses sequence π_2 during the processing of $P_{\sigma(i)}$ on M_1 and sequence π_1 during the processing of $P_{\sigma(i)}$ on M_2. We say that $P_{\sigma(i)}$ is processed using sequence π_{2-1}.

The following remark is required before we present our next result.

REMARK 6.1 The two-machine no-wait flow shop scheduling problem, denoted $F_2|\textit{no-wait}|C_t$, has as input a set of n jobs. Job $J_i, i = 1, ..., n$, is associated with two numerical parameters e_i and f_i (processing times on the two machines M_1' and M_2', respectively), and the objective is to find a job schedule ψ that minimizes $\sum_{i=1}^{n} \max\left\{e_{\psi(i+1)}, f_{\psi(i)}\right\}$. This problem can be solved in time $O(n \log n)$ by an algorithm due to Gilmore and Gomory [67]. See Appendix B for details. Since this algorithm is used several times in this chapter and the next, we simply refer to it as the Gilmore-Gomory algorithm.

LEMMA 6.2 *In general, an MPS sequence using the best CRM sequence is not optimal for* $RF_2|(free,A,MP,cyclic\text{-}n)|\mu$.

Proof. Consider the following instance of $RF_2|(free,A,MP,cyclic\text{-}n)|\mu$: $n = 5; k = 4; \delta_1 = 1, \delta_2 = 17, \delta_3 = 1; \epsilon_i = 1, i = 1, \ldots, 6; \eta = 1$; the processing times are shown in Table 6.1. Note that $\nu = 2\sum_{i=1}^{3} \delta_i + \epsilon_1 + \epsilon_2 + \epsilon_5 + \epsilon_6 - 2\eta = 40$.

From Lemma 6.1, the time to produce the parts in an MPS in the schedule σ under robot move sequence π_1 is

$$\begin{aligned} T_{1(\sigma)} &= \sum_{i=1}^{n} T^2_{1\sigma(i)\sigma(i+1)} = n\rho + \sum_{i=1}^{n}(a'_i + b'_i) \\ &= 5[1 + 1 + 34] + [10 + 400 + 10 + 400 + 10] \\ &\quad + [400 + 20 + 400 + 10 + 10] \\ &= 1850. \end{aligned}$$

The part scheduling problem under CRM sequence π_2 can be transformed into the problem $F_2|no\text{-}wait|C_t$ having the form: minimize $Z + \sum_{i=1}^{n} \max\{e_{\sigma(i+1)}, f_{\sigma(i)}\}$, where Z is a constant, $e_{\sigma(i)} = a'_{\sigma(i)}$, and $f_{\sigma(i)} = \max\{\nu, b'_{\sigma(i)}\}$ are the processing times for part $P_{\sigma(i)}$ on the first machine (M'_1) and the second machine (M'_2), respectively. In $F_2|no\text{-}wait|C_t$, the

i	1	2	3	4	5
a_i	7	397	7	397	7
b_i	397	17	397	7	7

Table 6.1. The Example Used in Lemma 6.2.

operations of any job are performed continuously without waiting on or between the machines (Remark 6.1 and Appendix B). Note that the schedule given by the Gilmore-Gomory algorithm is optimal if the shortest processing time on M'_1 is concurrent with that on M'_2, the second shortest processing time on M'_1 is concurrent with that on M'_2, and so on. For example, processing times e_2 and f_1 are concurrent in Figure 6.1, as are e_3 and f_2. If we use CRM sequence π_2, the part schedule $\{1, 2, 3, 4, 5\}$ satisfies this rule and is therefore optimal. From Lemma 6.1, the time to produce the parts in an MPS using schedule σ under CRM sequence

π_2 is

$$\begin{aligned} T_{2(\sigma)} &= \sum_{i=1}^{n} T^2_{2\sigma(i)\sigma(i+1)} \\ &= n\rho + \sum_{i=1}^{n} \max\{\nu, b'_{\sigma(i)}, a'_{\sigma(i+1)}\} \\ &= 180 + 400 + 40 + 400 + 40 + 40 \\ &= 1100. \end{aligned}$$

The use of a combination of different robot move sequences ($\pi_{1-2}, \pi_2, \pi_2, \pi_{2-1}, \pi_1$) in the two-machine flowshop using the same part schedule is illustrated in Figure 6.1. The part schedule is optimal for this combination, where part P_1 is processed under sequence π_{1-2}, parts P_2 and P_3 under sequence π_2, part P_4 under sequence π_{2-1}, and part P_5 under sequence π_1.

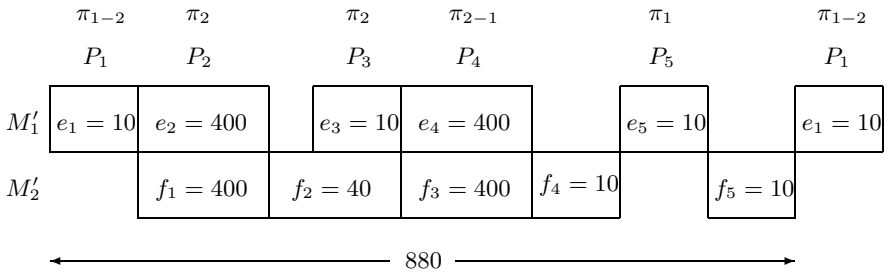


Figure 6.1. Combining Robot Move Sequences in Lemma 6.2.

The length of each operation is shown by its $e_{\sigma(i)}$ or $f_{\sigma(i)}$ value, and the contribution of the unavoidable robot travel time (i.e., $n\rho = 5(36) = 180$) is ignored in Figure 6.1. Note that $f_{\sigma(i)} = \max\{\nu, b'_{\sigma(i)}\}$ if part $P_{\sigma(i)}$ is processed by sequence π_2 or sequence π_{1-2}; otherwise $f_{\sigma(i)} = b'_{\sigma(i)}$. By adding in the travel time, we obtain a total cycle time of $880 + 180 = 1060$. Intuitively, we have taken advantage of the fact that some parts are suitable for processing under π_1 and others under π_2. This completes the proof of Lemma 6.2. ∎

REMARK 6.2 The procedure used by Sethi et al. [142] for this problem involves fixing the robot move sequence to π_1 and solving for the optimal part schedule, repeating this process for π_2, and then comparing the two

cycle times thus generated. Lemma 6.2 shows that the cycle time can be improved by using both sequences π_1 and π_2 in the repetitive production of an MPS. A further improvement for the above example can be achieved by using sequence π_{2-1} for part P_2 and sequence π_{1-2} for part P_3. The cycle time then becomes $870 + 180 = 1050$. This demonstrates that an optimal solution may need more than one transition from π_1 to π_2, and vice versa, in $RF_2|(free,A,MP,cyclic\text{-}n)|\mu$. Kise et al. [99] study the makespan version of the part scheduling problem under an Automated Guided Vehicle (AGV) cycle equivalent to π_2. Their problem is equivalent to $RF_2|(free,A,\ CRM(\pi_2))|C_{max}$, and they describe an $O(n^3)$ algorithm, where n is the number of jobs to be processed.

We now describe an algorithm, MinCycle, due to Hall et al. [75], that jointly optimizes the robot move sequence and the part schedule for problem $RF_2|(free,A,MP,cyclic\text{-}n)|\mu$. The idea behind the algorithm is as follows: MinCycle begins with an arbitrary part schedule using robot move sequence π_1 (Step 1) and by finding an optimal part schedule (from the Gilmore-Gomory algorithm) using robot move sequence π_2 (Step 4). It then compares their cycle times (in Step 2 of Subroutine GG). Between these two extreme solutions are many other solutions that use a mix of different robot move sequences; these are investigated in Step 5. In this step, the algorithm enumerates, in time $O(n^3)$, ways of setting q of the $a'_{\psi(i)}$ values and q of the b'_i values to a large number H, where $q = 1, \ldots, t$, $t = \min\{|\{i : a'_i < \nu\}|, |\{i : b'_i < \nu\}|\}$, $H = n\rho + \sum_{i=1}^{n}(a'_i + b'_i) + 1$, and the permutation ψ places the a_i's in ascending order. In Step 5, the $(q-1)$ smallest $a'_{\psi(i)}$'s and the $(q-1)$ smallest b'_i's are set to H. It remains to set one $a'_{\psi(i)}$ value and one b'_i value to H; all such possible assignments are examined. This requires time $O(n^2)$ for a fixed q. Each assignment of the q H's corresponds to a partition of the MPS into four subsets – B_1, B_2, B_{12}, and B_{21} – consisting of parts produced using robot move sequences π_1, π_2, π_{1-2}, and π_{2-1}, respectively. If a part P_i has both a'_i and b'_i set to H, then that part belongs to B_1. If only a'_i (resp., b'_i) is set to H, then that part belongs to B_{12}, (resp., B_{21}). If neither a'_i nor b'_i is set to H, then that part belongs to B_2. For each assignment of the q H's, Subroutine GG is called to solve the equivalent $F_2|no\text{-}wait|C_t$ problem. Following an optimality test (Step 2 of GG), the smallest cycle time C^*, and corresponding schedule σ^* with partitions

B_1, B_2, B_{12}, and B_{21} obtained from Steps 1, 4, and 5, are recorded as an optimal solution.

Note that since H is large, Subroutine GG provides a schedule in which each a'_i set to H is processed on M'_1 concurrently with a b'_j set to H on M'_2. This pairing in GG means that the processing times corresponding to these two H values, e_i and f_j, can be represented in the schedule as two consecutive blocks similar to f_4 and e_5 in Figure 6.1. Therefore, in any schedule, the sequence π_1 or π_{2-1} is followed by either π_1 or π_{1-2}. Similarly, the sequence π_2 or π_{1-2} is always followed by either π_2 or π_{2-1}. Step 2 of Subroutine GG calculates the actual cycle time of each partition defined by the assignment of the H values in Step 5 of MinCycle. The algorithm of Hall et al. [75] for $RF_2|(free,A,MP,cyclic\text{-}n)|\mu$ now follows.

Algorithm MinCycle

Step 0: Given $a_i, b_i, 1 \le i \le n, \delta_1, \delta_2, \delta_3, \eta$, and $\epsilon_1, \ldots, \epsilon_6$,
$B_1 = \{P_1, \ldots, P_n\}, B_2 = B_{12} = B_{21} = \emptyset$,
$\rho = \epsilon_3 + \epsilon_4 + 2\delta_2$,
$\nu = 2\sum_{i=1}^{3} \delta_i + \epsilon_1 + \epsilon_2 + \epsilon_5 + \epsilon_6 - 2\eta$.

Step 1: $a'_i = a_i + \epsilon_1 + \epsilon_2 + 2\delta_1 - \eta, i = 1, \ldots, n$,
$b'_i = b_i + \epsilon_5 + \epsilon_6 + 2\delta_3 - \eta, i = 1, \ldots, n$,
$\sigma^* = \{1, \ldots, n\}$,
$C^* = n\rho + \sum_{i=1}^{n}(a'_i + b'_i)$,
$H = C^* + 1$,
$Y = 0, q = 0$.

Step 2: Number the parts so that $b'_i \le \ldots \le b'_n$.
Find an ordering ψ of the parts such that $a'_{\psi(1)} \le \ldots \le a'_{\psi(n)}$.

Step 3: $t = \min\{|\{i : a'_i < \nu\}|, |\{i : b'_i < \nu\}|\}$.

Step 4: Call **GG**$(a'_i, b'_i, 1 \le i \le n)$.

Step 5: For $q = 1, \ldots, t$:
$Y = \sum_{i=1}^{q-1}(a'_{\psi(i)} + b'_i)$,
$a'_{\psi(i)} = H, b'_i = H, i = 1, \ldots, q-1$,
For $h, j = q, \ldots, n$:
$W_a = a'_{\psi(h)}, W_b = b'_j$,
$a'_{\psi(h)} = H, b'_j = H$,
$Y = Y + W_a + W_b$,
Call **GG**$(a'_i, b'_i, 1 \le i \le n)$,

$a'_{\psi(h)} = W_a, b'_j = W_b,$
$Y = Y - W_a - W_b.$
End
End

Subroutine GG$(a'_i, b'_i, 1 \le i \le n)$
Step 1: e_i $a'_i, f_i = \max\{\nu, b'_i\}, i = 1, \ldots, n.$
Apply the Gilmore-Gomory algorithm to the instance $\{e_i, f_i\}, i = 1, \ldots, n.$
Let σ denote the optimal part schedule, and C denote its value.
Step 2: If $C + n\rho - qH + Y < C^*$, then
$C^* = C + n\rho - qH + Y, \sigma^* = \sigma,$
$B_1 = \{P_i : a'_i = H\} \cap \{P_i : b'_i = H\},$
$B_{12} = \{P_i : a'_i = H\} \backslash \{P_i : b'_i = H\},$
$B_{21} = \{P_i : b'_i = H\} \backslash \{P_i : a'_i = H\},$
$B_2 = \{P_1, \ldots, P_n\} \backslash \{B_1 \cup B_{12} \cup B_{21}\}.$
End If

To prove the optimality of MinCycle, several preliminary results are required.

LEMMA 6.3 *In any feasible solution for* $RF_2|(free, A,\ MP, cyclic\text{-}n)|\mu$, *we have* $|B_{12}| = |B_{21}|$.

Proof. In general, any feasible solution is a cyclic production of an MPS in a schedule in which each part is produced using one of the following robot move sequences: π_1, π_2, π_{1-2}, and π_{2-1}. Note that sequence π_{1-2} must be preceded by π_1 or π_{2-1}, and followed by π_2 or π_{2-1}. Since the schedule is cyclic, it follows that the number of transitions from π_1 into π_2 (using π_{1-2}) must equal the number of transitions from π_2 into π_1 (using π_{2-1}). ∎

LEMMA 6.4 *Algorithm MinCycle evaluates all solutions in which* $|B_1 \cup B_{12}| = |B_1 \cup B_{21}| = q$, *where* $q = 0$ *if the solution is found in Step 4, and* $1 \le q \le t$ *if the solution is found in Step 5 and the* $q - 1$ *smallest* a'_i *(resp.,* b'_i*) values are associated with the parts in* $B_1 \cup B_{12}$ *(resp.,* $B_1 \cup B_{21}$*).*

Proof. The proof follows from Steps 4 and 5 of MinCycle and Step 2 of Subroutine GG. ∎

LEMMA 6.5 *For any real numbers* $v, w, x \geq 0$ *with* $w \geq v$, $\max\{x, v\} + w \geq \max\{x, w\} + v$.

Proof. $\max\{x, w\} = \max\{x, v + (w - v)\} \leq \max\{x, v\} + (w - v)$. ∎

THEOREM 6.1 *Algorithm MinCycle finds an optimal solution to* $RF_2|$ *(free,A,MP,cyclic-n)*$|\mu$ *in time* $O(n^4)$.

Proof. The first part of the proof shows that for given subsets B_1, B_2, B_{12}, and B_{21}, MinCycle finds an optimal part schedule σ whose cycle time is denoted by T_σ. As we will observe later, during the execution of this schedule, the robot may switch between robot move sequences π_1 and π_2. We write the cycle time as $T_\sigma = n\rho + \sum_{i=1}^{n} d_{\sigma(i)\sigma(i+1)}$, where we define $d_{\sigma(n)\sigma(n+1)} = d_{\sigma(n)\sigma(1)}$,

$$T_\sigma = n\rho + \sum_{i=1}^{n} V_{\sigma(i)\sigma(i+1)} + \sum_{i=1}^{n} W_{\sigma(i)\sigma(i+1)},$$

$$d_{\sigma(i)\sigma(i+1)} = V_{\sigma(i)\sigma(i+1)} = \begin{cases} b'_{\sigma(i)} + a'_{\sigma(i+1)} & \text{if } \pi_1 \text{ is used for the} \\ & \text{pair } (P_{\sigma(i)}, P_{\sigma(i+1)}); \\ 0 & \text{otherwise,} \end{cases}$$

$$d_{\sigma(i)\sigma(i+1)} = W_{\sigma(i)\sigma(i+1)} = \begin{cases} \max\{\nu, b'_{\sigma(i)}, a'_{\sigma(i+1)}\} & \text{if } \pi_2 \text{ is used for the} \\ & \text{pair } (P_{\sigma(i)}, P_{\sigma(i+1)}); \\ 0 & \text{otherwise.} \end{cases}$$

For these definitions, π_1 is used for the pair $(P_{\sigma(i)}, P_{\sigma(i+1)})$ if and only if $P_{\sigma(i)} \in B_1 \cup B_{21}$ and $P_{\sigma(i+1)} \in B_1 \cup B_{12}$. A similar characterization holds for π_2. Since $n\rho$ is a constant, the above problem is equivalent to that of minimizing

$$\bar{T}_\sigma = \sum_{i=1}^{n} V_{\sigma(i)\sigma(i+1)} + \sum_{i=1}^{n} W_{\sigma(i)\sigma(i+1)}.$$

Let $Y = \sum_{P_i \in B_1 \cup B_{12}} a'_i + \sum_{P_i \in B_1 \cup B_{21}} b'_i$. Then letting $b'_{\sigma(i)} = a'_{\sigma(i+1)} = H$, whenever $V_{\sigma(i)\sigma(i+1)} = b'_{\sigma(i)} + a'_{\sigma(i+1)}$, gives

$$\hat{T}_\sigma = \sum_{i=1}^{n} \max\{f_{\sigma(i)}, e_{\sigma(i+1)}\},$$

where $f_{\sigma(j)} = \max\{\nu, b'_{\sigma(j)}\}$ and $e_{\sigma(j)} = a'_{\sigma(j)}, j = 1, \ldots, n$. It remains to show that the schedule that minimizes $\hat{T}_\sigma$ also minimizes T_σ. To see this, note that since H is a large value, each a'_i set to H on M'_1 is concurrent with a b'_i set to H on M'_2 in any optimal solution. Thus, $qH \leq \hat{T}_{\sigma^*} < (q+1)H$. Step 2 of Subroutine GG ignores schedules σ for which $\hat{T}_\sigma \geq (q+1)H$, since the optimal solution can be found elsewhere. Thus, in σ^*, no part in B_{12} can immediately precede a part in B_1 or B_{12}, since this would imply that $\hat{T}_\sigma \geq (q+1)H$. Similarly, no part in B_{21} can immediately precede a part in B_2 or B_{21} in σ^*. Therefore, if $\hat{T}_\sigma < (q+1)H$, we have

$$\hat{T}_\sigma = T_\sigma - n\rho + qH - Y,$$

and the two problems are equivalent under this condition.

The second part of the proof shows that the partitions of the parts into subsets B_1, B_2, B_{12}, and B_{21} considered by MinCycle contain an optimal schedule. To characterize such partitions, let (ρ^*, σ^*) represent an optimal solution consisting of a partition ρ^* of the parts into B_1, B_2, B_{12}, and B_{21} and a part schedule σ^*. We make use of two claims.

- Claim 1. If $a'_{\sigma^*(i+1)} + b'_{\sigma^*(i)} < \nu$, then $d_{\sigma^*(i)\sigma^*(i+1)} = V_{\sigma^*(i)\sigma^*(i+1)}$, and if $a'_{\sigma^*(i+1)} + b'_{\sigma^*(i)} \geq \nu$, then $d_{\sigma^*(i)\sigma^*(i+1)} = W_{\sigma^*(i)\sigma^*(i+1)}$.
  Proof of Claim 1. Assume that there exist parts $P_{\sigma^*(i)}$ and $P_{\sigma^*(i+1)}$ such that

$$\begin{aligned} d_{\sigma^*(i)\sigma^*(i+1)} &= W_{\sigma^*(i)\sigma^*(i+1)}, \\ a'_{\sigma^*(i+1)} + b'_{\sigma^*(i)} &< \nu = \max\{\nu, b'_{\sigma^*(i)}, a'_{\sigma^*(i+1)}\} = W_{\sigma^*(i)\sigma^*(i+1)}. \end{aligned}$$

  Now, if we form a new solution (π', σ^*) by setting $d_{\sigma^*(i)\sigma^*(i+1)} = V_{\sigma^*(i)\sigma^*(i+1)} = b'_{\sigma^*(i)} + a'_{\sigma^*(i+1)}$, we find a new cycle time $T_{\pi'\sigma^*}$ satisfying $T_{\pi'\sigma^*} - T_{\pi^*\sigma^*} = b'_{\sigma^*(i)} + a'_{\sigma^*(i+1)} - \nu$. This implies $T_{\pi'\sigma^*} < T_{\pi^*\sigma^*}$, thus contradicting the optimality of (π^*, σ^*). Note that the change from π^* to π' alters the robot move sequences for parts $P_{\sigma^*(i)}$ and $P_{\sigma^*(i+1)}$. The proof of the second part is similar. □

- Claim 2. If there exists an optimal solution (π^*, σ^*) with $|B_1 \cup B_{12}| = |B_1 \cup B_{21}| = q$, then there exists an optimal solution satisfying the following two additional conditions:

(a) The $(q-1)$ smallest a'_i values are associated with parts in B_1 or B_{12}. Thus, $d_{\sigma^*(i-1)\sigma^*(i)} = V_{\sigma^*(i-1)\sigma^*(i)} = b'_{\sigma^*(i-1)} + a'_{\sigma^*(i)}$.

(b) The $(q-1)$ smallest b'_i values are associated with parts in B_1 or B_{21}. Thus, $d_{\sigma^*(i)\sigma^*(i+1)} = V_{\sigma^*(i)\sigma^*(i+1)} = b'_{\sigma^*(i)} + a'_{\sigma^*(i+1)}$.

Proof of Claim 2. For condition (a), assume that in (π^*, σ^*), part $P_{\sigma^*(j)} \in B_2 \cup B_{21}$, where $a'_{\sigma^*(j)}$ is one of the $(q-1)$ smallest a'_i values. It follows that there exist parts $P_{\sigma^*(u)}, P_{\sigma^*(v)} \in B_1 \cup B_{12}$ such that $a'_{\sigma^*(u)}, a'_{\sigma^*(v)} \geq a'_{\sigma^*(j)}$. Without loss of generality, let $\sigma^* = \{P_{\sigma^*(j)}, \sigma^*_{ju}, P_{\sigma^*(u)}, \sigma^*_{uv}, P_{\sigma^*(v)}, \sigma^*_{vj}\}$, where

$$\sigma^*_{ju} = \{P_{\sigma^*(j+1)}, P_{\sigma^*(j+2)}, \ldots, P_{\sigma^*(u-2)}, P_{\sigma^*(u-1)}\}$$

denotes a partial schedule of jobs in between the jth and uth positions of σ^*, as in Figure 6.2. Since parts $P_{\sigma^*(u)}, P_{\sigma^*(v)} \in B_1 \cup B_{12}$, it follows that parts $P_{\sigma^*(u-1)}, P_{\sigma^*(v-1)} \in B_1 \cup B_{21}$. Similarly, since part $P_{\sigma^*(j)} \in B_2 \cup B_{21}$, it follows that part $P_{\sigma^*(j-1)} \in B_2 \cup B_{12}$. Thus we can write

$$\begin{aligned} d_{\sigma^*(j-1)\sigma^*(j)} &= \max\{\nu, b'_{\sigma^*(j-1)}, a'_{\sigma^*(j)}\}, \\ d_{\sigma^*(u-1)\sigma^*(u)} &= b'_{\sigma^*(u-1)} + a'_{\sigma^*(u)}, \\ d_{\sigma^*(v-1)\sigma^*(v)} &= b'_{\sigma^*(v-1)} + a'_{\sigma^*(v)}. \end{aligned}$$

As shown in Figure 6.3, let (π', σ') be another solution with $\sigma' = \{P_{\sigma^*(u)}, \sigma^*_{uv}, P_{\sigma^*(j)}, \sigma^*_{ju}, P_{\sigma^*(v)}, \sigma^*_{vj}\}$. In σ', all parts belong to the same subsets as in σ^* with the following exceptions: $P_{\sigma^*(j)} \in B_1 \cup B_{12}$ and $P_{\sigma^*(u)} \in B_2 \cup B_{21}$. Thus in σ', we have

$$\begin{aligned} d_{\sigma^*(j-1)\sigma^*(u)} &= \max\{\nu, b'_{\sigma^*(j-1)}, a'_{\sigma^*(u)}\}, \\ d_{\sigma^*(v-1)\sigma^*(j)} &= b'_{\sigma^*(v-1)} + a'_{\sigma^*(j)}, \\ d_{\sigma^*(u-1)\sigma^*(v)} &= b'_{\sigma^*(u-1)} + a'_{\sigma^*(v)}. \end{aligned}$$

Comparing the cycle times of the two solutions, we have from Lemma 6.5,

$$\begin{aligned} T_{\pi'\sigma'} - T_{\pi^*\sigma^*} &= \max\{\nu, a'_{\sigma^*(u)}, b'_{\sigma^*(j-1)}\} + b'_{\sigma^*(v-1)} + a'_{\sigma^*(j)} \\ &\quad + b'_{\sigma^*(u-1)} + a'_{\sigma^*(v)} - \max\{\nu, a'_{\sigma^*(j)}, b'_{\sigma^*(j-1)}\} \\ &\quad - b'_{\sigma^*(u-1)} - a'_{\sigma^*(u)} - b'_{\sigma^*(v-1)} - a'_{\sigma^*(v)} \\ &= \max\{\nu, a'_{\sigma^*(u)}, b'_{\sigma^*(j-1)}\} + a'_{\sigma^*(j)} \\ &\quad - \max\{\nu, a'_{\sigma^*(j)}, b'_{\sigma^*(j-1)}\} - a'_{\sigma^*(u)} \\ &\leq 0, \end{aligned}$$

It follows that, without increasing the cycle time, we can repeat such changes until condition (a) is satisfied.

The proof for condition (b) is similar due to the following symmetry in the schedule. If we interchange the role of M'_1 and M'_2 and consider any schedule in the reverse direction, we find an equivalent schedule where b'_i (resp., a'_i) is the processing time on M'_1 (resp., M'_2). □

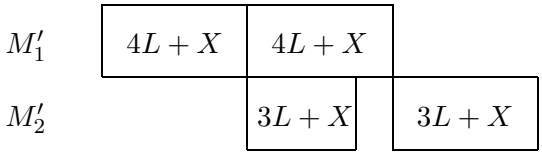


Figure 6.2. Partition π^* and Schedule $\sigma^* = \{P_{\sigma^*(j)}, \sigma^*_{ju}, P_{\sigma^*(u)}, \sigma^*_{uv}, P_{\sigma^*(v)}, \sigma^*_{vj}\}$ in Condition (a) of Theorem 6.1.

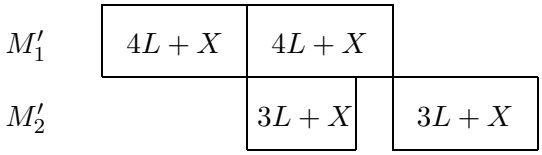


Figure 6.3. Partition π' and Schedule $\sigma' = \{P_{\sigma^*(u)}, \sigma^*_{uv}, P_{\sigma^*(j)}, \sigma^*_{ju}, P_{\sigma^*(v)}, \sigma^*_{vj}\}$ in Condition (a) of Theorem 6.1.

We are now able to prove the theorem. Steps 4 and 5 of MinCycle consider different values of q. If an optimal solution exists for a given value of q, Lemma 6.3 and Claim 2 define conditions present in at least

one optimal solution. Lemma 6.4 proves that all possible partitions of the parts using those conditions are examined in Step 4 for $q = 0$ and in Step 5 for $1 \leq q \leq t$. It follows from Claim 1 and the definition of t in Step 3 that the assignment of a value of H to any processing time of a part with index greater than t will not improve the cycle time. Thus, there exists an optimal solution with $0 \leq q \leq t$. Finally, given any partition, the first part of the proof establishes that Subroutine GG finds an optimal part schedule.

The dominant step in MinCycle is Step 5, where the indices h, j, and q are all $O(n)$. The number of calls of Subroutine GG is, therefore, $O(n^3)$. Lawler et al. [103] point out that Step 1 of Subroutine GG can be implemented in time $O(n \log n)$. However, the parts have already been ordered in Step 4 of MinCycle. Consequently, in Step 5, we only need to order the jobs whose processing times are set to a value of H. Since all such jobs have to appear at the end of the ordering in an arbitrary subschedule, the Gilmore-Gomory algorithm can be implemented in time $O(n)$. Step 2 of Subroutine GG also runs in time $O(n)$. Thus, the overall time complexity of MinCycle is $O(n^4)$. This completes the proof of Theorem 6.1. ∎

Algorithm MinCycle (Hall et al. [75]) has also been implemented in a two-machine robotic cell at the Manufacturing Systems Laboratory of the Industrial Engineering Department of the University of Toronto. The details of this implementation are provided by Chan and Lau [29]. Aneja and Kamoun [7] improve the complexity of MinCycle to $O(n \log n)$. We refer to this algorithm as Improved MinCycle.

6.3 Scheduling Multiple Part-Types in Three-Machine Cells

In this section, we study the problem of scheduling multiple part-types in a three-machine robotic cell under CRM sequences (problem $RF_3|(free,\ A,MP,CRM)|\mu$). CRM sequences are easy to understand and implement, and are therefore the focus of our attention here. We show in Section 6.3.2 that the optimal part scheduling problems associated with four of the six CRM sequences are polynomially solvable. The recognition version of the part scheduling problem is strongly NP-complete for the other two CRM sequences (Section 6.5). However, we do identify

several special cases for which the problem is polynomially solvable. The issue of convergence to a steady state is discussed in Section 6.4.

6.3.1 Cycle Time Derivations

Following Hall et al. [75], we now present the derivation of cycle times for CRM sequences $\pi_{1,3}, \pi_{2,3}, \ldots, \pi_{6,3}$, in $RF_3|(free,A,MP,CRM)|\mu$, starting from a convenient state in each case. Recall that $T^h_{j\sigma(i)\sigma(i+1)}$ denotes the time between the loading of part $P_{\sigma(i)}$ on machine M_h and the loading of part $P_{\sigma(i+1)}$ on machine M_h using CRM sequence $\pi_{j,3}$, where $h \in \{1,2,3\}$ and $j \in \{1,\ldots,6\}$.

Starting from the initial state $E = (\Omega, \emptyset, \emptyset, M_1^-)$, where the robot has just loaded part $P_{\sigma(i)}$ onto M_1 and where machines M_2 and M_3 are free, we can derive the following expression for $\pi_{1,3}$:

$$T^1_{1\sigma(i)\sigma(i+1)} = \alpha_1 + a_{\sigma(i)} + b_{\sigma(i)} + c_{\sigma(i)},$$

where $\alpha_1 = 2\sum_{i=1}^{4} \delta_i + \sum_{i=1}^{8} \epsilon_i - 3\eta$. We obtain

$$T_{1(\sigma)} = \sum_{i=1}^{n} T^1_{1\sigma(i+1)\sigma(i+2)} = n\alpha_1 + \sum_{i=1}^{n} (a_i + b_i + c_i).$$

Starting from the initial state $E = (\emptyset, \Omega, \Omega, M_2^-)$, where the robot has just loaded part $P_{\sigma(i)}$ onto machine M_2, M_3 is occupied with part $P_{\sigma(i-1)}$, and M_1 is free, we can derive the following expression for $\pi_{2,3}$:

$$T^2_{2\sigma(i)\sigma(i+1)} = \sum_{i=1}^{8} \epsilon_i + 2\delta_1 + 4\delta_2 + 4\delta_3 + 2\delta_4 - 4\eta + w_1^{i+1} + w_2^i + w_3^{i-1},$$

where

$$\begin{aligned}
w_1^{i+1} &= \max\{0, a_{2\sigma(i+1)} - \beta_2 - w_2^i\}, \\
w_2^i &= \max\{0, b_{2\sigma(i)} - \beta_2 - w_3^{i-1}\}, \\
w_3^{i-1} &= \max\{0, c_{2\sigma(i-1)} - \beta_2 - w_1^i\}, \\
\beta_2 &= \sum_{i=1}^{8} \epsilon_i + 2\delta_1 + 2\delta_2 + 2\delta_3 + 2\delta_4 - 3\eta.
\end{aligned}$$

Substituting for w_1^{i+1} and w_2^i, we obtain

$$\begin{aligned} T_{2(\sigma)} &= \sum_{i=1}^{n} T^2_{2\sigma(i)\sigma(i+1)} \\ &= n\alpha_2 + \sum_{i=1}^{n} \max\{\beta_2, a_{2\sigma(i+1)}, b_{2\sigma(i)} - w_3^{i-1}\} + \sum_{i=1}^{n} w_3^{i-1} \\ w_3^{i-1} &= \max\{0, c_{2\sigma(i-1)} - \max\{\beta_2, a_{2\sigma(i)} - \max\{0, b_{2\sigma(i-1)} \\ &\quad - \beta_2 - w_3^{i-2}\}\}\}, \end{aligned}$$

where $b_{2\sigma(n+1)} = b_{2\sigma(1)}$, $a_{2\sigma(n+1)} = a_{2\sigma(1)}$, $a_{2\sigma(n+2)} = a_{2\sigma(2)}$,

$$\begin{aligned} \alpha_2 &= 2\delta_2 + 2\delta_3 - \eta, \\ a_{2\sigma(i)} &= a_{\sigma(i)} + \sum_{i=1}^{4} \epsilon_i + \sum_{i=7}^{8} \epsilon_i + 2\delta_1 + 2\delta_4 - 2\eta, \\ b_{2\sigma(i)} &= b_{\sigma(i)} + \sum_{i=3}^{6} \epsilon_i \\ c_{2\sigma(i)} &= c_{\sigma(i)} + \sum_{i=1}^{2} \epsilon_i + \sum_{i=5}^{8} \epsilon_i + 2\delta_1 + 2\delta_4 - 2\eta. \end{aligned}$$

Starting from the initial state $E = (\emptyset, \emptyset, \Omega, M_3^-)$, where the robot has just loaded part $P_{\sigma(i)}$ onto machine M_3 and where M_1 and M_2 are free, we can derive the following expression for $\pi_{3,3}$:

$$\begin{aligned} T^3_{3\sigma(i)\sigma(i+1)} &= a_{\sigma(i+1)} + 2\delta_1 + 2\delta_2 + 4\delta_3 + 2\delta_4 + \sum_{i=1}^{8} \epsilon_i - 3\eta \\ &\quad + w_2^{i+1} + w_3^i, \text{ where} \\ w_2^{i+1} &= \max\{0, b_{\sigma(i+1)} - 2\delta_3 - 2\delta_4 - \epsilon_7 - \epsilon_8 + \eta - w_3^i\}, \\ w_3^i &= \max\{0, c_{\sigma(i)} - a_{\sigma(i+1)} - 2\delta_1 - 2\delta_2 - 2\delta_3 - \epsilon_1 - \epsilon_2 \\ &\qquad -\epsilon_3 - \epsilon_4 + 2\eta\}. \end{aligned}$$

This then implies

$$T^3_{3\sigma(i)\sigma(i+1)} = \alpha_3 + \max\{\beta_3 + a_{\sigma(i+1)}, b_{3\sigma(i+1)} + a_{\sigma(i+1)}, c_{3\sigma(i)}\}, \text{where}$$

$$\begin{aligned}
\alpha_3 &= 2\delta_3 + \epsilon_5 + \epsilon_6 - \eta, \\
\beta_3 &= 2\delta_1 + 2\delta_2 + 2\delta_3 + 2\delta_4 + \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 + \epsilon_7 + \epsilon_8 - 2\eta, \\
b_{3\sigma(i)} &= b_{\sigma(i)} + 2\delta_1 + 2\delta_2 + \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 - \eta, \\
c_{3\sigma(i)} &= c_{\sigma(i)} + 2\delta_4 + \epsilon_7 + \epsilon_8.
\end{aligned}$$

We thus obtain the cycle time $T_{3(\sigma)}$ as

$$\begin{aligned}
T_{3(\sigma)} &= \sum_{i=1}^{n} T^3_{3\sigma(i)\sigma(i+1)} \\
&= n\alpha_3 + \sum_{i=1}^{n} \max\{\beta_3 + a_{\sigma(i+1)}, b_{3\sigma(i+1)} + a_{\sigma(i+1)}, c_{3\sigma(i)}\}.
\end{aligned}$$

Starting from the initial state $E = (\emptyset, \emptyset, \Omega, M_3^-)$, where the robot has just loaded part $P_{\sigma(i)}$ onto machine M_3 and where M_1 and M_2 are free, we can derive the following expression for $\pi_{4,3}$:

$$\begin{aligned}
T^3_{4\sigma(i)\sigma(i+1)} &= b_{\sigma(i+1)} + 2\delta_1 + 4\delta_2 + 4\delta_3 + 2\delta_4 + \sum_{i=1}^{8} \epsilon_i - 5\eta \\
&\quad + w_1^{i+1} + w_3^i, \text{ where} \\
w_1^{i+1} &= \max\{0, a_{\sigma(i+1)} - w_3^i - 2\delta_2 - 2\delta_3 - 2\delta_4 - \epsilon_7 - \epsilon_8 + 3\eta\}, \\
w_3^i &= \max\{0, c_{\sigma(i)} - 2\delta_1 - 2\delta_2 - 2\delta_3 - \epsilon_1 - \epsilon_2 + 3\eta\}.
\end{aligned}$$

This gives

$$\begin{aligned}
T^3_{4\sigma(i)\sigma(i+1)} &= \alpha_4 + b_{\sigma(i+1)} + \max\{\beta_4, c_{4\sigma(i)}, a_{4\sigma(i+1)}\}, \text{ where} \\
\alpha_4 &= 2\delta_2 + 2\delta_3 + \epsilon_3 + \epsilon_4 + \epsilon_5 + \epsilon_6 - 2\eta, \\
\beta_4 &= 2\delta_1 + 2\delta_2 + 2\delta_3 + 2\delta_4 + \epsilon_1 + \epsilon_2 + \epsilon_7 + \epsilon_8 - 3\eta, \\
a_{4\sigma(i)} &= a_{\sigma(i)} + 2\delta_1 + \epsilon_1 + \epsilon_2, \\
c_{4\sigma(i)} &= c_{\sigma(i)} + 2\delta_4 + \epsilon_7 + \epsilon_8.
\end{aligned}$$

Thus, we obtain the cycle time $T_{4(\sigma)}$ as

$$T_{4(\sigma)} = \sum_{i=1}^{n} T^3_{4\sigma(i)\sigma(i+1)} = n\alpha_4 + \sum_{i=1}^{n} b_i + \sum_{i=1}^{n} \max\{\beta_4, c_{4\sigma(i)}, a_{4\sigma(i+1)}\}.$$

Starting from the initial state $E = (\emptyset, \Omega, \emptyset, M_2^-)$, where the robot has just loaded part $P_{\sigma(i)}$ onto machine M_2 and where M_1 and M_3 are free,

we can derive the following expression for $\pi_{5,3}$:

$$\begin{aligned} T^2_{5\sigma(i)\sigma(i+1)} &= c_{\sigma(i)} + 2\delta_1 + 4\delta_2 + 2\delta_3 + 2\delta_4 + \sum_{i=1}^{8} \epsilon_i - 3\eta + w_1^{i+1} + w_2^i, \\ \text{where} & \\ w_1^{i+1} &= \max\{0, a_{\sigma(i+1)} - w_2^i - 2\delta_2 - 2\delta_3 - 2\delta_4 - \epsilon_5 - \epsilon_6 \\ &\quad - \epsilon_7 - \epsilon_8 + 2\eta - c_{\sigma(i)}\}, \\ w_2^i &= \max\{0, b_{\sigma(i)} - 2\delta_1 - 2\delta_2 - \epsilon_1 - \epsilon_2 + \eta\}. \end{aligned}$$

This gives

$$\begin{aligned} T^2_{5\sigma(i)\sigma(i+1)} &= \alpha_5 + \max\{\beta_5 + c_{\sigma(i)}, b_{5\sigma(i)} + c_{\sigma(i)}, a_{5\sigma(i+1)}\}, \text{ where} \\ \alpha_5 &= 2\delta_2 + \epsilon_3 + \epsilon_4 - \eta, \\ \beta_5 &= 2\delta_1 + 2\delta_2 + 2\delta_3 + 2\delta_4 + \epsilon_1 + \epsilon_2 + \epsilon_5 + \epsilon_6 + \epsilon_7 + \epsilon_8 - 2\eta, \\ a_{5\sigma(i)} &= a_{\sigma(i)} + 2\delta_1 + \epsilon_1 + \epsilon_2, \\ b_{5\sigma(i)} &= b_{\sigma(i)} + 2\delta_3 + 2\delta_4 + \epsilon_5 + \epsilon_6 + \epsilon_7 + \epsilon_8 - \eta. \end{aligned}$$

Thus, we obtain the cycle time $T_{5(\sigma)}$ as

$$T_{5(\sigma)} = \sum_{i=1}^{n} T^2_{5\sigma(i)\sigma(i+1)} = n\alpha_5 + \sum_{i=1}^{n} \max\{\beta_5 + c_{\sigma(i)}, b_{5\sigma(i)} + c_{\sigma(i)}, a_{5\sigma(i+1)}\}.$$

Starting from the initial state $E = (\Omega, \Omega, \Omega, M_1^-)$, where the robot has just loaded part $P_{\sigma(i)}$ onto machine M_1 and where M_2 and M_3 are occupied by parts $P_{\sigma(i-1)}$ and $P_{\sigma(i-2)}$, respectively, we can derive the following expression for $\pi_{6,3}$:

$$\begin{aligned} T^1_{6\sigma(i)\sigma(i+1)} &= \sum_{i=1}^{8} \epsilon_i + 2\delta_1 + 4\delta_2 + 4\delta_3 + 2\delta_4 - 4\eta + w_1^i + w_2^{i-1} + w_3^{i-2}, \\ \text{where} & \\ w_1^i &= \max\{0, a_{6\sigma(i)} - \beta_6 - w_2^{i-1} - w_3^{i-2}\}, \\ w_2^{i-1} &= \max\{0, b_{6\sigma(i-1)} - \beta_6 - w_3^{i-2}\}, \\ w_3^{i-2} &= \max\{0, c_{6\sigma(i-2)} - \beta_6 - w_1^{i-1}\}. \end{aligned}$$

Substituting for w_1^i, w_2^{i-1} and w_3^{i-2}, we obtain

$$\begin{aligned} T_{6(\sigma)} &= \sum_{i=1}^{n} T^1_{6\sigma(i)\sigma(i+1)} \\ &= n\alpha_6 + \sum_{i=1}^{n} \max\{\beta_6, c_{6\sigma(i-2)} - w_1^{i-1}, b_{6\sigma(i-1)}, a_{6\sigma(i)}\}, \end{aligned}$$

where

$$\begin{aligned} w_1^{i-1} &= \max\{0, a_{6\sigma(i-1)} - \max\{\beta_6, b_{6\sigma(i-2)}, c_{6\sigma(i-3)} - w_1^{i-2}\}\}, \\ \alpha_6 &= 0; \quad \beta_6 = 2\delta_1 + 4\delta_2 + 4\delta_3 + 2\delta_4 + \sum_{i=1}^{8} \epsilon_i - 4\eta; \\ a_{6\sigma(i)} &= a_{\sigma(i)} + 2\delta_1 + 2\delta_2 + \sum_{i=1}^{4} \epsilon_i - \eta, \\ b_{6\sigma(i)} &= b_{\sigma(i)} + 2\delta_2 + 2\delta_3 + \sum_{i=3}^{6} \epsilon_i - \eta, \\ c_{6\sigma(i)} &= c_{\sigma(i)} + 2\delta_3 + 2\delta_4 + \sum_{i=5}^{8} \epsilon_i - \eta. \end{aligned}$$

6.3.2 Efficiently Solvable Special Cases

We now study the cycle time minimization problem under CRM sequences π_1, π_3, π_4, and π_5. Whenever appropriate, we let $a_{\sigma(n+1)} = a_{\sigma(1)}$ and $a_{\sigma(n+2)} = a_{\sigma(2)}$; similar definitions hold for $b_{\sigma(n+1)}, b_{\sigma(n+2)}, c_{\sigma(n+1)}$, and $c_{\sigma(n+2)}$. We also let $w_i^{n+1} = w_i^1$ for $i = 1, 2, 3$.

THEOREM 6.2 *Problem $RF_3|(free,A,MP,CRM(\pi_1))|\mu$ can be solved trivially.*

Proof. From Section 6.3.1, we have

$$T_{1(\sigma)} = \sum_{i=1}^{n} T^1_{1\sigma(i)\sigma(i+1)} = n\alpha_1 + \sum_{i=1}^{n} (a_i + b_i + c_i).$$

Clearly, $T_{1(\sigma)}$ depends only on $a_i, b_i, c_i, 1 \le i \le n, \alpha_1$ and n, and is independent of the schedule of parts. ∎

In Theorems 6.3, 6.4, and 6.5, we show that the cycle time of the part scheduling problem under CRM sequences π_3, π_4, and π_5, respectively, can be expressed in the form $C + \sum_{i=1}^{n} \max\{e_{\sigma(i+1)}, f_{\sigma(i)}\}$ for some con-

stant C. A schedule that minimizes the cycle time can then be obtained in time $O(n \log n)$ by the Gilmore-Gomory algorithm (See Appendix B).

THEOREM 6.3 *Problem $RF_3|(free,A,MP,CRM(\pi_3))|\mu$ can be solved optimally in time $O(n \log n)$.*

Proof. From Section 6.3.1 we have

$$T^3_{3\sigma(i)\sigma(i+1)} = \alpha_3 + \max\{\beta_3 + a_{\sigma(i+1)}, b_{3\sigma(i+1)} + a_{\sigma(i+1)}, c_{3\sigma(i)}\}.$$

Letting $e_{3\sigma(i)} = \max\{\beta_3 + a_{\sigma(i)}, b_{3\sigma(i)} + a_{\sigma(i)}\}$, we have

$$T_{3(\sigma)} = \sum_{i=1}^{n} T^3_{3\sigma(i)\sigma(i+1)} = n\alpha_3 + \sum_{i=1}^{n} \max\{e_{3\sigma(i+1)}, c_{3\sigma(i)}\}.$$

■

THEOREM 6.4 *Problem $RF_3|(free,A,MP,CRM(\pi_4))|\mu$ can be solved optimally in time $O(n \log n)$.*

Proof. From Section 6.3.1, we have

$$T^3_{4\sigma(i)\sigma(i+1)} = \alpha_4 + b_{\sigma(i+1)} + \max\{\beta_4, c_{4\sigma(i)}, a_{4\sigma(i+1)}\}.$$

Letting $e_{4\sigma(i)} = \max\{\beta_4, a_{4\sigma(i)}\}$, we have

$$T_{4(\sigma)} = \sum_{i=1}^{n} T^3_{4\sigma(i)\sigma(i+1)} = n\alpha_4 + \sum_{i=1}^{n} b_{\sigma(i)} + \sum_{i=1}^{n} \max\{e_{4\sigma(i+1)}, c_{4\sigma(i)}\}.$$

■

THEOREM 6.5 *Problem $RF_3|(free,A,MP,CRM(\pi_5))|\mu$ can be solved optimally in time $O(n \log n)$.*

Proof. From Section 6.3.1, we have

$$T^2_{5\sigma(i)\sigma(i+1)} = \alpha_5 + \max\{\beta_5 + c_{\sigma(i)}, b_{5\sigma(i)} + c_{\sigma(i)}, a_{5\sigma(i+1)}\}.$$

Letting $f_{5\sigma(i)} = \max\{\beta_5 + c_{\sigma(i)}, b_{5\sigma(i)} + c_{\sigma(i)}\}$, we have

$$T_{5(\sigma)} = \sum_{i=1}^{n} T^2_{5\sigma(i)\sigma(i+1)} = n\alpha_5 + \sum_{i=1}^{n} \max\{a_{5\sigma(i+1)}, f_{5\sigma(i)}\}.$$

■

To derive conditions under which problem $RF_3|(free,A,\ MP,\ CRM)|\mu$ is polynomially solvable, we need the following preliminary results. We begin by characterizing the cycle time for parts $P_{\sigma(1)},\ldots,P_{\sigma(n)}$, under CRM sequence π_2, where σ denotes a schedule of the parts.

LEMMA 6.6 *The cycle time in $RF_3|(free,A,MP,CRM(\pi_2))|\mu$ for the production of n parts in schedule σ under CRM sequence π_2 is given by*

$$T_{2(\sigma)} = n\alpha_2 + \sum_{i=1}^{n} \max\{\beta_2, a_{2\sigma(i+2)}, b_{2\sigma(i+1)} - w_3^i\} + \sum_{i=1}^{n} w_3^i,$$

where

$$w_3^i = \max\{0, c_{2\sigma(i)} - \max\{\beta_2, a_{2\sigma(i+1)} - \max\{0, b_{2\sigma(i)} - \beta_2 - w_3^{i-1}\}\}\}.$$

Proof. See Section 6.3.1. ∎

REMARK 6.3 When $\delta_i = \delta, i = 1,\ldots,4, \epsilon_i = \epsilon, i = 1,\ldots,8$, and $\eta = 0$, the cycle time in $RF_3|(free,A,MP,CRM(\pi_2))|\mu$ can be obtained from Lemma 6.6 as follows:

$$\begin{aligned}
T_{2(\sigma)} &= n\alpha + \bar{T}_\sigma, \text{ where} \\
\bar{T}_\sigma &= \sum_{i=1}^{n} \max\{\beta, a_{\sigma(i+2)} + \frac{\beta}{2}, b_{\sigma(i+1)} - w_3^i\} + \sum_{i=1}^{n} w_3^i, \\
w_3^i &= \max\{0, c_{\sigma(i)} - \max\{\frac{\beta}{2}, a_{\sigma(i+1)} - \max\{0, b_{\sigma(i)} - \beta - w_3^{i-1}\}\}\}.
\end{aligned}$$

In the definition of $w_3^i, \alpha = 4\delta + 4\epsilon$ and $\beta = 8\delta + 4\epsilon$.

We next characterize the production cycle time for CRM sequence π_6.

LEMMA 6.7 *The cycle time in $RF_3|(free,A,MP,CRM(\pi_6))|\mu$ for the production of n parts in schedule σ under CRM sequence π_6 is given by*

$$T_{6(\sigma)} = n\alpha_6 + \sum_{i=1}^{n} \max\{\beta_6, c_{6\sigma(i)} - w_1^{i+1}, b_{6\sigma(i+1)}, a_{6\sigma(i+2)}\},$$

where

$$w_1^{i+1} = \max\{0, a_{6\sigma(i+1)} - \max\{\beta_6, b_{6\sigma(i)}, c_{6\sigma(i-1)} - w_1^i\}\}.$$

Proof. See Section 6.3.1. ∎

REMARK 6.4 When $\delta_i = \delta, i = 1, \ldots, 4, \epsilon_i = \epsilon, i = 1, \ldots, 8$, and $\eta = 0$, the cycle time in $RF_3|(free,A,MP,CRM(\pi_6))|\mu$ can be obtained from Lemma 6.7 as follows:

$$\begin{aligned} T_{6(\sigma)} &= n\alpha + \bar{T}_\sigma, \text{ where} \\ \bar{T}_\sigma &= \sum_{i=1}^{n} \max\{\beta, c_{\sigma(i)} - w_1^{i+1}, b_{\sigma(i+1)}, a_{\sigma(i+2)}\}, \\ w_1^{i+1} &= \max\{0, a_{\sigma(i+1)} - \max\{\beta, b_{\sigma(i)}, c_{\sigma(i-1)} - w_1^i\}\}. \end{aligned}$$

In the definition of w_1^{i+1}, $\alpha = 4\delta + 4\epsilon$ and $\beta = 8\delta + 4\epsilon$.

LEMMA 6.8 *In $RF_3|(free,A,MP,CRM)|\mu$,*

$$\min\{T_{2(\sigma)}, T_{6(\sigma)}\} \geq n(2\delta_1 + 4\delta_2 + 4\delta_3 + 2\delta_4 + \sum_{i=1}^{8} \epsilon_i - 4\eta).$$

Proof. Using π_2 as described in Section 6.3.1,

$$T_{2(\sigma)} \geq n(\alpha_2 + \beta_2) = n(2\delta_1 + 4\delta_2 + 4\delta_3 + 2\delta_4 + \sum_{i=1}^{8} \epsilon_i - 4\eta).$$

Similarly, using π_6 as described in Section 6.3.1,

$$T_{6(\sigma)} \geq n(\alpha_6 + \beta_6) = n(2\delta_1 + 4\delta_2 + 4\delta_3 + 2\delta_4 + \sum_{i=1}^{8} \epsilon_i - 4\eta).$$

∎

It follows from Theorems 6.2, 6.3, 6.4, and 6.5 that problem $RF_3|(free, A,MP,CRM)|\mu$ is polynomially solvable in two situations. The first of these occurs when the data are such that the best schedules under π_2 and π_6 can be found in polynomial time. The second situation occurs when the cycle time provided by the best schedule under π_2 or π_6 cannot be optimal. In the theorem that follows, condition A corresponds to the first situation, and conditions B through E to the second.

THEOREM 6.6 *Problem $RF_3|(free, A, MP, CRM)\mu$ can be solved optimally in time $O(n \log n)$ under any of the conditions A, B, C, D, or E*

below:

$$
\begin{aligned}
A: \quad & \delta_i = \delta, i = 0, \ldots, 4; \epsilon_i = \epsilon, i = 1, \ldots, 8; \eta = 0; \\
& c_i \leq 4\delta + 2\epsilon, i = 1, \ldots, n. \\
B: \quad & \sum_{i=1}^{n} (a_i + b_i + c_i) \leq n(2\delta_2 + 2\delta_3 - \eta). \\
C1: \quad & a_i \leq 2\delta_2 - \eta, i = 1, \ldots, n. \\
C2: \quad & b_i \leq 2\delta_3 + 2\delta_4 + \epsilon_7 + \epsilon_8 - \eta, i = 1, \ldots, n. \\
C3: \quad & c_i \leq 2\delta_1 + 4\delta_2 + 2\delta_3 + \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 - 3\eta, i = 1, \ldots, n. \\
D1: \quad & a_i \leq 2\delta_2 + 2\delta_3 + 2\delta_4 + \epsilon_7 + \epsilon_8 - 3\eta, i = 1, \ldots, n. \\
D2: \quad & b_i \leq \eta, i = 1, \ldots, n. \\
D3: \quad & c_i \leq 2\delta_1 + 2\delta_2 + 2\delta_3 + \epsilon_1 + \epsilon_2 - 3\eta, i = 1, \ldots, n. \\
E1: \quad & a_i \leq 2\delta_2 + 4\delta_3 + 2\delta_4 + \epsilon_5 + \epsilon_6 + \epsilon_7 + \epsilon_8 - 3\eta, i = 1, \ldots, n. \\
E2: \quad & b_i \leq 2\delta_1 + 2\delta_2 + \epsilon_1 + \epsilon_2 - \eta, i = 1, \ldots, n. \\
E3: \quad & c_i \leq 2\delta_3 - \eta, i = 1, \ldots, n.
\end{aligned}
$$

Proof. Consider each condition A through E in turn.

A: From Remark 6.3 and the conditions in A, we have $T_{2(\sigma)} = n(4\delta + 4\epsilon) + \sum_{i=1}^{n} \max\{f_{\sigma(i)}, b_{\sigma(i+1)}\}$, where $f_{\sigma(i)} = \max\{a_{\sigma(i+2)} + 4\delta + 2\epsilon, 8\delta + 4\epsilon\}$. It now follows that the part scheduling problem can be solved by the Gilmore-Gomory algorithm in time $O(n \log n)$. Similarly, from Remark 6.4 and the conditions in A, we have $T_{6(\sigma)} = n(4\delta + 4\epsilon) + \sum_{i=1}^{n} \max\{8\delta + 4\epsilon, b_{\sigma(i+1)}, a_{\sigma(i+2)}\}$, with the same result. It then follows from Theorems 6.2, 6.3, 6.4, and 6.5 that the optimal part schedules under all six CRM sequences can be found, and their cycle times compared, in time $O(n \log n)$.

B: $T_{1(\sigma)} = \sum_{i=1}^{n} (a_i + b_i + c_i) + n(2\sum_{i=1}^{4} \delta_i + \sum_{i=1}^{8} \epsilon_i - 3\eta) \leq n(2\delta_2 + 2\delta_3 - \eta) + n(2\sum_{i=1}^{4} \delta_i + \sum_{i=1}^{8} \epsilon_i - 3\eta)$ from condition B, and $T_{1(\sigma)} \leq \min\{T_{2(\sigma)}, T_{6(\sigma)}\}$ from Lemma 6.8.

C: From Theorem 6.3 and Lemma 6.8,
$T_{3(\sigma)} = \sum_{i=1}^{n} \max\{\alpha_3 + \beta_3 + a_{\sigma(i+1)}, \alpha_3 + b_{3\sigma(i+1)} + a_{\sigma(i+1)}, \alpha_3 + c_{3\sigma(i)}\}$
$\leq \sum_{i=1}^{n} \max\{(2\delta_3 + \epsilon_5 + \epsilon_6 - \eta) + (2\delta_1 + 2\delta_2 + 2\delta_3 + 2\delta_4 + \epsilon_1 + \epsilon_2$

$+\ \epsilon_3+\epsilon_4+\epsilon_7+\epsilon_8-2\eta)+(2\delta_2-\eta),(2\delta_3+\epsilon_5+\epsilon_6-\eta)+(2\delta_3+2\delta_4$
$+\ \epsilon_7+\epsilon_8-\eta+2\delta_1+2\delta_2+\epsilon_1+\epsilon_2+\epsilon_3+\epsilon_4-\eta)+(2\delta_2-\eta),(2\delta_3+\epsilon_5$
$+\ \epsilon_6-\eta)+(2\delta_1+4\delta_2+2\delta_3+\epsilon_1+\epsilon_2+\epsilon_3+\epsilon_4-3\eta+2\delta_4+\epsilon_7+\epsilon_8)\}$
$=\sum_{i=1}^{n}\max\{2\delta_1+4\delta_2+4\delta_3+2\delta_4+\sum_{i=1}^{8}\epsilon_i-4\eta,2\delta_1+4\delta_2+4\delta_3+$
$2\delta_4+\sum_{i=1}^{8}\epsilon_i-4\eta,2\delta_1+4\delta_2+4\delta_3+2\delta_4+\sum_{i=1}^{8}\epsilon_i-4\eta\}$
$\leq\min\{T_{2(\sigma)},T_{6(\sigma)}\}.$

D: The proof is similar to part C except that we now use Theorem 6.4.

E: The proof is similar to part C except that we now use Theorem 6.5.

■

REMARK 6.5 The results in Theorem 6.6 provide considerable intuition about tradeoffs between different CRM sequences. Specifically, each of the conditions in Theorem 6.6 suggests that if processing times are short relative to robot travel times, then it is better to wait at a machine while it is processing as CRM sequences π_1, π_3, π_4, and π_5 do, rather than to load the part and move elsewhere as CRM sequences π_2 and π_6 do.

In Section 6.5, we show that the decision versions of the part scheduling problems under CRM sequences π_2 and π_6 are strongly NP-complete.

6.4 Steady-State Analyses

Note that all the results in this book relate to the performance of robotic cells operating in a steady state. Thus, a question to be asked is whether the cell must be initialized into its steady state. Alternatively, what happens if it starts from a state that does not occur in its steady state cycle? We show here that a three-machine cell reaches a steady state in a number of cycles that is bounded by a function of the cell data. Similar results can be proved for larger cells, and this section's results can easily be applied to cells producing identical parts. Usually the production of parts begins from an initial state E_I, where the cell is empty and the robot is at I ready to pick up a part. We first show how a cell converges to a steady state starting from state E_I under CRM sequences $\pi_1, \ldots, \pi_6$ in RF_3. It is easy to see that the following remark concerning a steady state for cells processing multiple part-types is consistent with the definition in Chapter 3.

REMARK 6.6 If the cell operates in a steady state then the waiting times of the robot, for each part on each machine, are identical for each MPS.

In CRM sequences π_1, π_3, π_4, and π_5, there is a state E_q^* in which the robot has just completed the loading of a part onto machine M_q and all other machines are free. In π_1, that machine can be M_1, M_2, or M_3; in π_3, the machine is M_3; in π_4, the machine can be M_2 or M_3; and in π_5 the machine is M_2. In every case, E_q^* can be reached from the state E_I within the production of one MPS. Therefore, the process of convergence to a steady state is very simple in these four CRM sequences. For the same reason, CRM sequences π_1 and π_2 in RF_2 converge to a steady state within the production of one MPS from the initial starting state E_I.

REMARK 6.7 In RF_3, the state E_q^* effectively decomposes production into separate MPSs in cycles based on CRM sequences π_1, π_3, π_4, and π_5. Thus, a single MPS may be analyzed by itself. This is one of the properties that makes the part scheduling problem using these CRM sequences solvable in polynomial time, as discussed in Section 6.3.2.

The process of convergence to a steady state using CRM sequences π_2 and π_6 remains to be considered.

6.4.1 Reaching Steady State for the Sequence $CRM(\pi_2)$

We first analyze CRM sequence π_2 for which the cycle time expression is derived in Lemma 6.6. First, we need two preliminary results.

LEMMA 6.9 *Let $w_3^1, \ldots, w_3^n$ and $w_3'^1, \ldots, w_3'^n$ denote two feasible vectors of waiting times in π_2, where*

$$\begin{aligned} w_3^{i+1} &= \max\{0, c_{2\sigma(i+1)} \\ &\quad - \max\{\beta_2, a_{2\sigma(i+2)} - \max\{0, b_{2\sigma(i+1)} - w_3^i - \beta_2\}\}\}, \\ w_3'^{i+1} &= \max\{0, c_{2\sigma(i+1)} \\ &\quad - \max\{\beta_2, a_{2\sigma(i+2)} - \max\{0, b_{2\sigma(i+1)} - w_3'^i - \beta_2\}\}\}. \end{aligned}$$

If $w_3^i \le w_3'^i$, then $w_3'^{i+1} \le w_3^{i+1}$ and $w_3^{i+2} \le w_3'^{i+2}$.

Proof. Since $w_3'^i \ge w_3^i$, $\max\{0, b_{2\sigma(i+1)} - w_3'^i - \beta_2\} \le \max\{0, b_{2\sigma(i+1)} - w_3^i - \beta_2\}$. This implies that $\max\{\beta_2, a_{2\sigma(i+2)} - \max\{0, b_{2\sigma(i+1)} -$

$w_3'^i - \beta_2\}\} \geq \max\{\beta_2, a_{2\sigma(i+2)} - \max\{0, b_{2\sigma(i+1)} - w_3^i - \beta_2\}\}$. Thus, $\max\{0, c_{2\sigma(i+1)} - \max\{\beta_2, a_{2\sigma(i+2)} - \max\{0, b_{2\sigma(i+1)} - w_3'^i - \beta_2\}\}\} \leq \max\{0, c_{2\sigma(i+1)} - \max\{\beta_2, a_{2\sigma(i+2)} - \max\{0, b_{2\sigma(i+1)} - w_3^i - \beta_2\}\}\}$. We therefore have $w_3'^{i+1} \leq w_3^{i+1}$. The second part of the proof follows from the first part by setting $i = i + 1$. ∎

Owing to the behavior of the waiting times w_3^i in Lemma 6.9, we need the following lemma to define a steady state for the MPS cycle corresponding to $CRM(\pi_2)$.

LEMMA 6.10 *For n even, the MPS cycle with $CRM(\pi_2)$ consisting of a single MPS defines a steady state. However, for n odd, two MPSs may be required to define a steady state for the MPS cycle with $CRM(\pi_2)$.*

Proof. Let $w_j^i(q)$ denote the waiting time of the robot at machine M_j for part $P_{\sigma(i)}$ in the qth iteration of the cycle. If n is even, it is clear from the recursive relations for $w_3^i(q)$ and Lemma 6.9 that if $w_3^i(q) \leq w_3'^i(q)$, then $w_3^i(q+1) \leq w_3'^i(q+1)$. Thus, a cycle containing one MPS is sufficient to define a steady state. If n is odd, it is similarly clear that if $w_3^i(q) \leq w_3'^i(q)$, then $w_3^i(q+1) \geq w_3'^i(q+1)$. However, $w_3^i(q+2) \leq w_3'^i(q+2)$. Therefore, two MPSs may be required to define a steady state. ∎

EXAMPLE 6.1 $n = 3$; $a_{2\sigma(1)} = 5, b_{2\sigma(1)} = 3, c_{2\sigma(1)} = 7$; $a_{2\sigma(2)} = 6, b_{2\sigma(2)} = 8, c_{2\sigma(2)} = 8$; $a_{2\sigma(3)} = 8, b_{2\sigma(3)} = 7, c_{2\sigma(3)} = 4$, where $\sigma = (P_1, P_2, P_3)$, and $\beta_2 = 1$.

Consider the schedule $\sigma = (P_1, P_2, P_3)$. The steady-state waiting times at machine M_3 for the first, second, and third MPSs are computed as follows:

For the first MPS, starting with $w_3^3(0) = 0$,

$w_3^1(1) = \max\{0, c_{2\sigma(1)} - \max\{\beta_2, a_{2\sigma(2)} - \max\{0, b_{2\sigma(1)} - \beta_2 - w_3^3(0)\}\}\} = 3,$
$w_3^2(1) = \max\{0, c_{2\sigma(2)} - \max\{\beta_2, a_{2\sigma(3)} - \max\{0, b_{2\sigma(2)} - \beta_2 - w_3^1(1)\}\}\} = 4,$
$w_3^3(1) = \max\{0, c_{2\sigma(3)} - \max\{\beta_2, a_{2\sigma(1)} - \max\{0, b_{2\sigma(3)} - \beta_2 - w_3^2(1)\}\}\} = 1.$

For the second MPS,

$w_3^1(2) = \max\{0, c_{2\sigma(1)} - \max\{\beta_2, a_{2\sigma(2)} - \max\{0, b_{2\sigma(1)} - \beta_2 - w_3^3(1)\}\}\} = 2,$
$w_3^2(2) = \max\{0, c_{2\sigma(2)} - \max\{\beta_2, a_{2\sigma(3)} - \max\{0, b_{2\sigma(2)} - \beta_2 - w_3^1(2)\}\}\} = 5,$
$w_3^3(2) = \max\{0, c_{2\sigma(3)} - \max\{\beta_2, a_{2\sigma(1)} - \max\{0, b_{2\sigma(3)} - \beta_2 - w_3^2(2)\}\}\} = 0.$

For the third MPS,

$w_3^1(3) = \max\{0, c_{2\sigma(1)} - \max\{\beta_2, a_{2\sigma(2)} - \max\{0, b_{2\sigma(1)} - \beta_2 - w_3^3(2)\}\}\} = 3,$

$w_3^2(3) = \max\{0, c_{2\sigma(2)} - \max\{\beta_2, a_{2\sigma(3)} - \max\{0, b_{2\sigma(2)} - \beta_2 - w_3^1(3)\}\}\} = 4,$
$w_3^3(3) = \max\{0, c_{2\sigma(3)} - \max\{\beta_2, a_{2\sigma(1)} - \max\{0, b_{2\sigma(3)} - \beta_2 - w_3^2(3)\}\}\} = 1.$

Note that the first set of waiting times of parts at M_3 in the first MPS is different from that in the second MPS and is equal to that in the third MPS. Therefore, it is necessary in this case to include two MPSs to define a cycle in a steady state.

In view of Lemma 6.10, we let v denote the number of parts in a cycle, where $v = 2n$ if n is odd and $v = n$ if n is even. For n odd, a given part schedule σ means a schedule of $2n$ parts in which the first n parts corresponding to an MPS have the same order as the last n parts. Thus, we renumber the parts in a cycle as $P_{\sigma(1)}, \ldots, P_{\sigma(n)}, P_{\sigma(n+1)}, \ldots, P_{\sigma(2n)}$, where $P_{\sigma(i)}$ and $P_{\sigma(n+i)}$ are the same part type for $1 \le i \le n$.

The idea behind the algorithm is as follows. If there exists a steady-state solution that satisfies certain lower and upper bounds on the waiting times, then the recursive expressions for w_3^i in Lemma 6.6 can be greatly simplified, and one such solution is found in Step 1. Alternatively, at least one of the lower or upper bound is violated, in which case one waiting time value is fixed, and such a solution is found in Step 2. The algorithm consists of a series of tests that check which of these conditions is satisfied.

Algorithm FindTime2
Input: α_2, β_2 and $a_{2\sigma(j)}, b_{2\sigma(j)}, c_{2\sigma(j)}, j = 1, \ldots, n$.
Step 1: $LB^i = \max\{0, b_{2\sigma(i+1)} - a_{2\sigma(i+2)}\}, i = 1, \ldots, v.$
$UB^i = \min\{b_{2\sigma(i+1)} - \beta_2, c_{2\sigma(i+1)} + b_{2\sigma(i+1)} - a_{2\sigma(i+2)} - \beta_2\}, i = 1, \ldots, v.$
Test1: If $UB^i < 0$ for some $i, 1 \le i \le v$, then go to Step 2.
Test2: If $LB^i > UB^i$ for some $i, 1 \le i \le v$, then go to Step 2.
$Wait(i) = LB^i$, $i = 1, \ldots, v$.
Test3:
For $j = 1, \ldots, v$, do
$w_3^j = Wait(j)$.
$w_3^{j+i} = c_{2\sigma(j+i)} - a_{2\sigma(j+i+1)} + b_{2\sigma(j+i)} - \beta_2 - w_3^{j+i-1}$, $i = 1, \ldots, v$.
If $w_3^j = w_3^{j+v}$ and $LB^i \le w_3^{j+i} \le UB^i, i = 1, \ldots, v$, then
$\frac{v}{n} T_{2(\sigma)} = v\alpha_2 + \sum_{i=1}^{v} (a_{2\sigma(i)} + b_{2\sigma(i)} + c_{2\sigma(i)} - \beta_2)/2$.
Terminate.
End If
End
Set $Wait(i) = UB^i$, $i = 1, \ldots, v$, and repeat **Test3**.

Step 2: $Wait(i) = \max\{0, c_{2\sigma(i)} - \max\{\beta_2, a_{2\sigma(i+1)}\}\}$, $i = 1, \ldots, v$.

Test4:

For $j = 1, \ldots, v$, do

$w_3^j = Wait(j)$.

For $i = 1, \ldots, v$, find

$w_3^{j+i} = \max\{0, x\}$ where

$x = c_{2\sigma(j+i)} - \max\{\beta_2, a_{2\sigma(j+i+1)} - \max\{0, b_{2\sigma(j+i)} - w_3^{j+i-1} - \beta_2\}\}$.

If $w_3^j = w_3^{j+v}$, then

$\frac{v}{n} T_{2(\sigma)} = v\alpha_2 + \sum_{i=1}^{v} \max\{\beta_2, b_{2\sigma(i+1)} - w_3^i, a_{2\sigma(i+2)}\} + \sum_{i=1}^{v} w_3^i$.

Terminate.

End If

End

Set $Wait(i) = \max\{0, c_{2\sigma(i)} - \beta_2\}$, $i = 1, \ldots, v$, and repeat **Test4**.

Set $Wait(i) = 0$, $i = 1, \ldots, v$, and repeat **Test4**.

To discuss the optimality of FindTime2, we need the following preliminary results.

LEMMA 6.11 *If there exists a feasible MPS cycle with $CRM(\pi_2)$ using part schedule σ and having cycle time $T_{2(\sigma)}$, and if in this cycle (a) $w_3^i = c_{2\sigma(i)} - a_{2\sigma(i+1)} + b_{2\sigma(i)} - \beta_2 - w_3^{i-1}$ and (b) $LB^i \le w_3^i \le UB^i, i = 1, \ldots, v$, in steady state, then there exist either one or two feasible MPS cycles with $CRM(\pi_2)$ that have cycle time $T_{2(\sigma)}$ and (i) $LB^i \le w_3^i \le UB^i, i = 1, \ldots, v$, where $w_3^h = LB^h$ or $w_3^h = UB^h$ in one cycle, and (ii) $w_3^j = LB^j$ or $w_3^j = UB^j$ in the other cycle (if it exists) for some $1 \le h, j \le v$.*

Proof. Let $w_3^1, \ldots, w_3^v$ denote a feasible vector of waiting times using part schedule σ, and let $\Delta_1 = \min_{1 \le i \le v}\{w_3^1 - LB^1, UB^2 - w_3^2, \ldots, w_3^{v-1} - LB^{v-1}, UB^v - w_3^v\}$, $h = \text{argmin}_{1 \le i \le n}\{w_3^1 - LB^1, UB^2 - w_3^2, \ldots, w_3^{v-1} - LB^{v-1}, UB^v - w_3^v\}$. Let $x^1 = w_3^1 - \Delta_1$, $x^2 = w_3^2 + \Delta_1, \ldots, x^{v-1} = w_3^{v-1} - \Delta_1$, $x^v = w_3^v + \Delta_1$. The new vector of waiting times x^i, $i = 1, \ldots, v$, also satisfies the equations in the lemma, and we have $x^h = LB^h$ or $x^h = UB^h$, depending on whether h is odd or even. Similarly, let $\Delta_2 = \min_{1 \le i \le v}\{UB^1 - w_3^1, w_3^2 - LB^2, \ldots, UB^{v-1} - w_3^{v-1}, w_3^v - LB^v\}$, $j = \text{argmin}_{1 \le i \le n}\{UB^1 - w_3^1, w_3^2 - LB^2, \ldots, UB^{v-1} - w_3^{v-1}, w_3^v - LB^v\}$. Let $y^1 = w_3^1 + \Delta_2$, $y^2 = w_3^2 - \Delta_2, \ldots, y^{v-1} = w_3^{v-1} + \Delta_2$, $y^v = w_3^v - \Delta_2$. The new vector of waiting times y^i, $i = 1, \ldots, v$, also satisfies the equations in the lemma, and we have $x^j = LB^j$ or $x^j = UB^j$, as above. Since $T_{2(\sigma)}$ is independent of $w_3^i, i = 1, \ldots, v$, under the conditions stated in the lemma, $T_{2(\sigma)}$ is unchanged. ∎

REMARK 6.8 If $w_3^i = x^i, i = 1, \ldots, v$, and $w_3^i = y^i, i = 1, \ldots, v$, are the solutions from two feasible cycles that satisfy the conditions of Lemma 6.11, then $x^i \neq y^i$, $i = 1, \ldots, v$. Without loss of generality, assume that $x^1 < y^1$. Then there exists an $\eta > 0$ such that $y^1 = x^1 + \eta$, $y^2 = x^2 - \eta$, ..., $y^{v-1} = x^{v-1} + \eta$, and $y^v = x^v - \eta$, where $\eta = \min_{1 \leq i \leq n}\{UB^1 - x^1, x^2 - LB^2, \ldots, UB^{v-1} - x^{v-1}, x^v - LB^v\}$ is also a feasible solution. All the solutions in between x^i and y^i, $i = 1, \ldots, v$, satisfy $w_3^i = c_{2\sigma(i)} - a_{2\sigma(i+1)} + b_{2\sigma(i)} - \beta_2 - w_3^{i-1}$ and $LB^i \leq w_3^i \leq UB^i, i = 1, \ldots, v$. Thus, an infinite number of solutions with the same cycle time exists in this case. We note that Step 1 finds only one solution, which is x^i or $y^i, i = 1, \ldots, v$. However, given $x^i, i = 1, \ldots, v$, it is easy to determine y^i, and vice versa. If there exists only one feasible solution, then $\eta = 0$ and $x^i = y^i, i = 1, \ldots, v$.

LEMMA 6.12 *There cannot exist both a feasible vector of waiting times found under the conditions of Step 1 of FindTime2 with cycle time $\bar{T}$ and a feasible vector of waiting times found under the conditions of Step 2 of FindTime2 with cycle time $\hat{T}$, where $\bar{T} \neq \hat{T}$.*

Proof. Suppose that a feasible vector of waiting times $w_3^i = x^i, i = 1, \ldots, v$, is found in Step 1, and another feasible vector of waiting times $w_3^i = z^i, i = 1, \ldots, v$, is found in Step 2. Note that x^i,$i = 1, \ldots, v$, satisfies the conditions of Step 1. Let y^i,$i = 1, \ldots, v$, be another feasible vector of waiting times that satisfies the conditions of Step 1. If there exists some $j, 1 \leq j \leq v$, where $x^j = z^j$, then from Lemma 6.6, $x^i = z^i, i = 1, \ldots, v$. Thus $x^i \neq z^i$ and $y^i \neq z^i$, $i = 1, \ldots, v$. Without loss of generality, we may assume that $x^1 < y^1$. Thus, we have three cases: (1) $x^1 < z^1 < y^1$, (2) $x^1 > z^1$, and (3) $z^1 > y^1$. If $x^1 < z^1 < y^1$, then $LB^i < z^i < UB^i, i = 1, \ldots, v$, and $z^i, i = 1, \ldots, v$, cannot be found in Step 2 as claimed. If $x^1 > z^1$, then from Lemma 6.11 we may assume without loss of generality that $x^h = LB^h$ or $x^h = UB^h$. If $x^h = LB^h$, then $z^h < LB^h$, and from Lemma 6.6, $x^{h+1} = z^{h+1}$, which is a contradiction. If $x^h = UB^h$, then $z^h > UB^h$, and from Lemma 6.6, again we have $x^{h+1} = z^{h+1}$, which is a contradiction. Similarly, if $z^1 > y^1$, then from Lemma 6.11 we may assume without loss of generality that $y^j = LB^j$ or $y^j = UB^j$. If $y^j = LB^j$, then $z^j < LB^j$ and, from Lemma 6.6, $y^{j+1} = z^{j+1}$, which is a contradiction. If $y^j = UB^j$, then $z^j > UB^j$ and, from Lemma 6.6, we have $y^{j+1} = z^{j+1}$, which is a contradiction. ∎

Initialization of MPS cycle with $CRM(\pi_2)$: There are two different ways in which the MPS cycle with $CRM(\pi_2)$ can be initialized from the state E_I in $RF_3|(free,A,MP,\ CRM(\pi_2))|\mu$. Let $w_j^i(0)$ denote the waiting time of the robot at machine M_j for part $P_{\sigma(i)}$ in the initialization step.

(a) Initialization INA2, starting with part $P_{\sigma(i)}$: pick up $P_{\sigma(i)}$ (ϵ_1), move to M_1 (δ_1), load $P_{\sigma(i)}$ onto M_1 (ϵ_2), wait at M_1 ($w_1^i(0) = a_{\sigma(i)}$), unload $P_{\sigma(i)}$ from M_1 (ϵ_3), move to M_2 (δ_2), load $P_{\sigma(i)}$ onto M_2 (ϵ_4), move to I ($\delta_1 + \delta_2 - \eta$), pick up $P_{\sigma(i+1)}$ (ϵ_1), move to M_1 (δ_1), load $P_{\sigma(i+1)}$ onto M_1 (ϵ_2), move to M_2 (δ_2), wait (if necessary) at M_2 ($w_2^i(0)$), unload $P_{\sigma(i)}$ from M_2 (ϵ_5), move to M_3 (δ_3), load $P_{\sigma(i)}$ onto M_3 (ϵ_6), move to M_1 ($\delta_2+\delta_3-\eta$), wait (if necessary) at M_1 ($w_1^{i+1}(0)$), unload $P_{\sigma(i+1)}$ from M_1 (ϵ_3), move to M_2 (δ_2), and load $P_{\sigma(i+1)}$ onto M_2 (ϵ_4). At this point, the MPS cycle with $CRM(\pi_2)$ starts from the state $E = (\emptyset, \Omega, \Omega, M_2^-)$.

The waiting time equations corresponding to initialization INA2 are as follows:

$$\begin{aligned}
w_1^i(0) &= a_{\sigma(i)}, \\
w_2^i(0) &= \max\{0, b_{\sigma(i)} - 2\delta_1 - 2\delta_2 - \epsilon_1 - \epsilon_2 + \eta\}, \\
w_1^{i+1}(0) &= \max\{0, a_{\sigma(i+1)} - w_2^i(0) - 2\delta_2 - 2\delta_3 - \epsilon_5 - \epsilon_6 + \eta\}, \\
w_3^i(0) &= \max\{0, c_{\sigma(i)} - w_1^{i+1}(0) - 2\delta_2 - 2\delta_3 - \epsilon_3 - \epsilon_4 + \eta\},
\end{aligned}$$

The waiting time equations corresponding to the qth iteration of the MPS cycle with $CRM(\pi_2)$ are as follows:

$$\begin{aligned}
w_3^j(q) &= \max\{0, c_{2\sigma(j)} - \max\{\beta_2, a_{2\sigma(j+1)} \\
&\quad - \max\{0, b_{2\sigma(j)} - \beta_2 - w_3^{j-1}(q-1)\}\}\}, j = i+1, \\
w_3^j(q) &= \max\{0, c_{2\sigma(j)} - \max\{\beta_2, a_{2\sigma(j+1)} \\
&\quad - \max\{0, b_{2\sigma(j)} - \beta_2 - w_3^{j-1}(q)\}\}\}, j = i+2, \ldots, i+v,
\end{aligned}$$

where $w_j^i(q)$ denotes the waiting time of the robot at machine M_j for part $P_{\sigma(i)}$ in the qth cycle, $q \geq 1$, $w_3^{j+v}(q) = w_3^j(q), j = 1, \ldots, v$, and $P_{\sigma(i)}$ is the first part produced, starting from E_I.

(b) Initialization INB2, starting with part $P_{\sigma(i)}$: pick up $P_{\sigma(i)}$ (ϵ_1), move to M_1 (δ_1), load $P_{\sigma(i)}$ onto M_1 (ϵ_2), wait at M_1 ($w_1^i(0) = a_{\sigma(i)}$), unload $P_{\sigma(i)}$ from M_1 (ϵ_3), move to M_2 (δ_2), load $P_{\sigma(i)}$ onto M_2 (ϵ_4), wait at M_2 ($w_2^i(0) = b_{\sigma(i)}$), unload $P_{\sigma(i)}$ from M_2 (ϵ_5), move to M_3 (δ_3), load $P_{\sigma(i)}$ onto M_3 (ϵ_6), move to I ($\delta_1+\delta_2+\delta_3-2\eta$), pick up $P_{\sigma(i+1)}$ (ϵ_1), move

to M_1 (δ_1), load $P_{\sigma(i+1)}$ onto M_1 (ϵ_2), wait at M_1 ($w_1^{i+1}(0) = a_{\sigma(i+1)}$), unload $P_{\sigma(i+1)}$ from M_1 (ϵ_3), move to M_2 (δ_2), and load $P_{\sigma(i+1)}$ onto M_2 (ϵ_4). At this point, the MPS cycle with $CRM(\pi_2)$ starts from the state $E = (\emptyset, \Omega, \Omega, M_2^-)$.

The waiting time equations corresponding to initialization INB2 are as follows: $w_1^i(0) = a_{\sigma(i)}, w_2^i(0) = b_{\sigma(i)}, w_1^{i+1}(0) = a_{\sigma(i+1)}$, and $w_3^i(0) = \max\{0, c_{\sigma(i)} - w_1^{i+1}(0) - 2\delta_1 - 2\delta_2 - 2\delta_3 - \epsilon_1 - \epsilon_2 - \epsilon_3 - \epsilon_4 + 2\eta\}$. The waiting time equations corresponding to the qth iteration of the MPS cycle with $CRM(\pi_2)$ are as follows:

$$\begin{aligned} w_3^j(q) &= \max\{0, c_{2\sigma(j)} - \max\{\beta_2, a_{2\sigma(j+1)} \\ &\quad - \max\{0, b_{2\sigma(j)} - \beta_2 - w_3^{j-1}(q-1)\}\}\}, j = i+1, \\ w_3^j(q) &= \max\{0, c_{2\sigma(j)} - \max\{\beta_2, a_{2\sigma(j+1)} \\ &\quad - \max\{0, b_{2\sigma(j)} - \beta_2 - w_3^{j-1}(q)\}\}\}, j = i+2, \ldots, i+v, \end{aligned}$$

where $q = 1, 2, \ldots$, and $w_3^{j+v}(q) = w_3^j(q), j = 1, \ldots, v$.

The system can be initialized in two different ways, INA2 and INB2, and in each case there are n different possible parts in σ to start with. Thus, there are $2n$ possible ways to initialize the system. The following theorem uses one of them to provide an upper bound on the number of MPS cycles needed to reach a steady state.

THEOREM 6.7 *Starting from the state E_I and using part schedule σ, the cell in $RF_3|$(free,A,MP,$CRM(\pi_2)$)$|\mu$ will go through at most* $\max\{1, \min_{1\le i\le v}\{s_i\}\}$ *cycles before reaching a steady state, where* $s_i = c_{2\sigma(i)} - \max\{\beta_2, a_{2\sigma(i+1)} - \max\{0, b_{2\sigma(i+1)} - \beta_2\}\}$, $i = 1, 2, \ldots, v$, *and this bound is attainable.*

Proof. Assume, without loss of generality, that all the parameters are integers. The waiting time equations in Lemma 6.6 show that once the same value of w_3^i occurs in two consecutive cycles for some $i, 1 \le i \le v$, a steady state has been reached. An induction argument, based on Lemma 6.9, shows that $w_3^i(q)$ monotonically increases or decreases as q increases, until a steady state is reached. We have $w_3^i = 0$ if $s_i \le 0$ for some i, $1 \le i \le v$. In this case, the system reaches a steady state in the first cycle. On the other hand, we have $0 \le w_3^i \le s_i$ if $s_i > 0$ for $i = 1, \ldots, v$. In this case, it takes at most $\min_{1\le i\le v}\{s_i\}$ number of cycles before the system reaches a steady state. ∎

EXAMPLE 6.2 $n = 4; \epsilon_i = 0, i = 1, \ldots, 8; \delta_i = 0, i = 1, \ldots, 4; \eta = 0; \beta_2 = 0$. We let $a_{2\sigma(1)} = X, b_{2\sigma(1)} = X, c_{2\sigma(1)} = X$; $a_{2\sigma(2)} = X, b_{2\sigma(2)} = X, c_{2\sigma(2)} = X + 1$; $a_{2\sigma(3)} = X, b_{2\sigma(3)} = X, c_{2\sigma(3)} = X$; and $a_{2\sigma(4)} = X, b_{2\sigma(4)} = X, c_{2\sigma(4)} = X$, where $\sigma = (P_1, P_2, P_3, P_4)$ and $X > 0$ is an integer. We use initialization INA2, starting with part $P_{\sigma(1)}$ in schedule σ. Then, we have

$$
\begin{aligned}
w_1^1(0) &= a_{\sigma(1)} = X, \\
w_2^1(0) &= \max\{0, b_{2\sigma(1)} - 2\delta_1 - 2\delta_2 - \epsilon_1 - \epsilon_2 + \eta\} = X, \\
w_1^2(0) &= \max\{0, a_{2\sigma(2)} - w_2^1(0) - 2\delta_2 - 2\delta_3 - \epsilon_5 - \epsilon_6 + \eta\} = 0, \\
w_3^1(0) &= \max\{0, c_{2\sigma(1)} - w_1^2(0) - 2\delta_2 - 2\delta_3 - \epsilon_3 - \epsilon_4 + \eta\} = X, \\
w_3^2(1) &= \max\{0, c_{2\sigma(2)} - \max\{\beta_2, a_{2\sigma(3)} \\
&\qquad - \max\{0, b_{2\sigma(2)} - \beta_2 - w_3^1(0)\}\}\} = 1, \\
w_3^3(1) &= \max\{0, c_{2\sigma(3)} - \max\{\beta_2, a_{2\sigma(4)} \\
&\qquad - \max\{0, b_{2\sigma(3)} - \beta_2 - w_3^2(1)\}\}\} = X - 1, \\
w_3^4(1) &= \max\{0, c_{2\sigma(4)} - \max\{\beta_2, a_{2\sigma(1)} \\
&\qquad - \max\{0, b_{2\sigma(4)} - \beta_2 - w_3^3(1)\}\}\} = 1, \\
w_3^1(1) &= \max\{0, c_{2\sigma(1)} - \max\{\beta_2, a_{2\sigma(2)} \\
&\qquad - \max\{0, b_{2\sigma(1)} - \beta_2 - w_3^4(1)\}\}\} = X - 1, \\
w_3^2(2) &= \max\{0, c_{2\sigma(2)} - \max\{\beta_2, a_{2\sigma(3)} \\
&\qquad - \max\{0, b_{2\sigma(2)} - \beta_2 - w_3^1(1)\}\}\} = 2, \\
w_3^3(2) &= \max\{0, c_{2\sigma(3)} - \max\{\beta_2, a_{2\sigma(4)} \\
&\qquad - \max\{0, b_{2\sigma(3)} - \beta_2 - w_3^2(2)\}\}\} = X - 2, \\
w_3^4(2) &= \max\{0, c_{2\sigma(4)} - \max\{\beta_2, a_{2\sigma(1)} \\
&\qquad - \max\{0, b_{2\sigma(4)} - \beta_2 - w_3^3(2)\}\}\} = 2, \\
w_3^1(2) &= \max\{0, c_{2\sigma(1)} - \max\{\beta_2, a_{2\sigma(2)} \\
&\qquad - \max\{0, b_{2\sigma(1)} - \beta_2 - w_3^4(2)\}\}\} = X - 2.
\end{aligned}
$$

Thus, $w_3^2(q)$ (resp., $w_3^4(q)$) keeps increasing by one time unit per MPS cycle until it reaches $X + 1$ (resp., X), at which time a steady state has also been reached. In this case, it takes X cycles to reach a steady state.

6.4.2 Reaching Steady State for the Sequence $CRM(\pi_6)$

We first analyze CRM sequence π_6 for which the cycle time expression is derived in Lemma 6.7. We start with the following preliminary result.

LEMMA 6.13 *Let* $w_1^1, \ldots, w_1^n$, *and* $w_1'^1, \ldots, w_1'^n$, *denote two feasible vectors of waiting times in* π_6, *where* $w_1^{i+1} = \max\{0, a_{6\sigma(i+1)} - \max\{\beta_6, b_{6\sigma(i)}, c_{6\sigma(i-1)} - w_1^i\}\}$ *and* $w_1'^{i+1} = \max\{0, a_{6\sigma(i+1)} - \max\{\beta_6, b_{6\sigma(i)}, c_{6\sigma(i-1)} - w_1'^i\}\}$. *If* $w_1^i \leq w_1'^i$, *then* $w_1^{i+1} \leq w_1'^{i+1}$.

Proof. The proof is similar to that of Lemma 6.9.

REMARK 6.9 It follows from Lemma 6.13 that, whether n is even or odd, the MPS cycle with $CRM(\pi_6)$ consisting of a single MPS is sufficient to define a steady state.

We now show how the cycle time $T_{6(\sigma)}$ can be calculated in polynomial time for a given part schedule σ (Hall et al. [76]). The following algorithm delivers the correct waiting times and cycle time for the MPS cycle with $CRM(\pi_6)$.

Algorithm FindTime6
Input: α_6, β_6 and $a_{6\sigma(j)}, b_{6\sigma(j)}, c_{6\sigma(j)}, j = 1, \ldots, n$.
Step 1: $LB^i = \max\{0, c_{6\sigma(i-1)} - a_{6\sigma(i+1)}\}, i = 1, \ldots, n$,
$UB^i = \min\{c_{6\sigma(i-1)} - \beta_6, c_{6\sigma(i-1)} - b_{6\sigma(i)}\}, i = 1, \ldots, n$,
Test1: if $UB^i < 0$ for any $i = 1, \ldots, n$, then go to Step 2.
Test2: if $LB^i > UB^i$ for any $i = 1, \ldots, n$, then go to Step 2,
$Wait(i) = LB^i$, $i = 1, \ldots, n$.
Test3:
For $j = 1, \ldots, n$, do
$w_1^j = Wait(j)$,
$w_1^{j+i} = a_{6\sigma(j+i)} - c_{6\sigma(j+i-2)} + w_1^{j+i-1}$, $i = 1, \ldots, n$,
If $w_1^j = w_1^{j+n}$ and $LB^i \leq w_1^{j+i} \leq UB^i, i = 1, \ldots, n$, then
$T_{6(\sigma)} = n\alpha_6 + \sum_{i=1}^{n} a_{6i}$,
Terminate.
End If
End
Step 2: $Wait(i) = \max\{0, a_{6\sigma(i)} - \max\{\beta_6, b_{6\sigma(i-1)}\}\}$, $i = 1, \ldots, n$.
Test4 :
For $j = 1, \ldots, n$ do
$w_1^j = Wait(j)$,

For $i = 1, \ldots, n$, find
$w_1^{j+i} = \max\{0, a_{6\sigma(j+i)} - \max\{\beta_6, b_{6\sigma(j+i-1)}, c_{6\sigma(j+i-2)} - w_1^{j+i-1}\}\}$,
If $w_1^j = w_1^{j+n}$, then
$T_{6(\sigma)} = n\alpha_6 + \sum_{i=1}^{n} \max\{\beta_6, c_{6\sigma(i)} - w_1^{i+1}, b_{6\sigma(i+1)}, a_{6\sigma(i+2)}\}$,
Terminate.
End If
End
Set $Wait(i) = 0$, $i = 1, \ldots, n$, and repeat **Test4**.

To discuss the optimality of FindTime6, we need the following preliminary results.

LEMMA 6.14 *If there exists a feasible MPS cycle with $CRM(\pi_6)$ using part schedule σ and having cycle time $T_{6(\sigma)}$ in which (a) $w_1^i = a_{6\sigma(i)} - c_{6\sigma(i-2)} + w_1^{i-1}$ and (b) $LB^i \le w_1^i \le UB^i, i = 1, \ldots, n$, in steady state, then there exist either one or two feasible MPS cycles with $CRM(\pi_6)$ that have cycle time $T_{6(\sigma)}$ and (i) $LB^i \le w_1^i \le UB^i, i = 1, \ldots, n$, where $w_1^h = LB^h$ in the first cycle, and (ii) $w_1^j = UB^j$ in the second cycle (if it exists) for some $1 \le h, j \le n$.*

Proof. Let $w_1^1, \ldots, w_1^n$, denote a feasible vector of waiting times using part schedule σ. Let $\Delta_1 = \min_{1\le i\le n}\{w_1^i - LB^i\}$, $h = \text{argmin}_{1\le i\le n}\{w_1^i - LB^i\}$, and let $x^i = w_1^i - \Delta_1$, $i = 1, \ldots, n$. The new vector of waiting times x^i, $i = 1, \ldots, n$, also satisfies the equations in the lemma and $x^h = LB^h$. Similarly, let $\Delta_2 = \min_{1\le i\le n}\{UB^i - w_1^i\}$ and $j = \text{argmin}_{1\le i\le n}\{UB^i - w_1^i\}$. Let $y^i = w_1^i + \Delta_2$, $i = 1, \ldots, n$. The new vector of waiting times y^i, $i = 1, \ldots, n$, also satisfies the equations in the lemma and $y^j = UB^j$. Since $T_{6(\sigma)}$ is independent of $w_1^i, i = 1, \ldots, n$, under the conditions stated in the lemma, $T_{6(\sigma)}$ is unchanged. ∎

REMARK 6.10 If $w_1^i = x^i, i = 1, \ldots, n$, and $w_1^i = y^i, i = 1, \ldots, n$, are the solutions from two feasible cycles that satisfy the conditions of Lemma 6.14, then $x^i \ne y^i, i = 1, \ldots, n$, and there exists a $\xi > 0$ such that $y^i = x^i + \xi$, where $\xi = \min_{1\le i\le n}\{UB^i - x^i\}$. All the solutions $w_1^i = x^i + \Delta$, $i = 1, \ldots, n$, with $0 \le \Delta \le \xi$, satisfy $w_1^i = a_{6\sigma(i)} - c_{6\sigma(i-2)} + w_1^{i-1}$ and $LB^i \le w_1^i \le UB^i, i = 1, \ldots, n$. Thus, an infinite number of solutions with the same cycle time exist in this case. We note that Step 1 finds only one solution x^i, $i = 1, \ldots, n$. However, in view of $y^i = x^i + \xi$, if x^i is given, then $y^i, i = 1, \ldots, n$, can easily be determined. If there exists only one feasible solution, then $\xi = 0$ and $x^i = y^i, i = 1, \ldots, n$.

LEMMA 6.15 *There cannot exist two distinct feasible vectors of waiting times, one under the conditions of Step 1 and the other under the conditions of Step 2 of FindTime6, such that their corresponding cycle times are unequal.*

Proof. Suppose a feasible vector of waiting times $w_1^i = x^i, i = 1, \ldots, n$, is found in Step 1, and another feasible vector of waiting times $w_1^i = z^i, i = 1, \ldots, n$, is found in Step 2. Note that x^i,$i = 1, \ldots, n$, satisfies the conditions of Step 1. Let $y^i, i = 1, \ldots, n$, be another feasible vector of waiting times that satisfies the conditions of Step 1. If there exists some $j, 1 \leq j \leq n$, where $x^j = z^j$, then from Lemma 6.7, $x^i = z^i, i = 1, \ldots, n$. Thus, $x^i \neq z^i$ and $y^i \neq z^i$, $i = 1, \ldots, n$. From Lemma 6.13 we have three cases: (1) $x^i < z^i < y^i$, $i = 1, \ldots, n$, (2) $x^i > z^i$, $i = 1, \ldots, n$, and (3) $z^i > y^i$, $i = 1, \ldots, n$. If $x^i < z^i < y^i$, $i = 1, \ldots, n$, then $LB^i < z^i < UB^i, i = 1, \ldots, n$. Thus, $z^i, i = 1, \ldots, n$, cannot be found in Step 2 as claimed. If $x^i > z^i$, $i = 1, \ldots, n$, then from Lemma 6.14 we may assume, without loss of generality, that $x^h = LB^h$. If $x^h = LB^h$, then $z^h < LB^h$, and from Lemma 6.7, $x^{h+1} = z^{h+1}$, which provides a contradiction. Similarly, if $z^i > y^i$, $i = 1, \ldots, n$, then from Lemma 6.14, we may assume without loss of generality that $y^j = UB^j$. If $y^j = UB^j$, then $z^j > UB^j$, and from Lemma 6.7, $y^{j+1} = z^{j+1}$, which provides a contradiction. ∎

Initialization of an MPS cycle with $CRM(\pi_6)$: There are two different ways in which an MPS cycle with $CRM(\pi_6)$ can be initialized from the state E_I in $RF_3|(free,A,MP,\ CRM(\pi_6))|\mu$:

(a) Initialization INA6, starting with part $P_{\sigma(i)}$: pick up $P_{\sigma(i)}$ (ϵ_1), move to M_1 (δ_1), load $P_{\sigma(i)}$ onto M_1 (ϵ_2), wait at M_1 ($w_1^i(0) = a_{\sigma(i)}$), unload $P_{\sigma(i)}$ from M_1 (ϵ_3), move to M_2 (δ_2), load $P_{\sigma(i)}$ onto M_2 (ϵ_4), move to I ($\delta_1 + \delta_2 - \eta$), pick up $P_{\sigma(i+1)}$ (ϵ_1), move to M_1 (δ_1), load $P_{\sigma(i+1)}$ onto M_1 (ϵ_2), move to M_2 (δ_2), wait (if necessary) at M_2 ($w_2^i(0)$), unload $P_{\sigma(i)}$ from M_2 (ϵ_5), move to M_3 (δ_3), load $P_{\sigma(i)}$ onto M_3 (ϵ_6), move to M_1 ($\delta_2 + \delta_3 - \eta$), wait (if necessary) at M_1 ($w_1^{i+1}(0)$), unload $P_{\sigma(i+1)}$ from M_1 (ϵ_3), move to M_2 (δ_2), load $P_{\sigma(i+1)}$ onto M_2 (ϵ_4), move to I ($\delta_1 + \delta_2 - \eta$), pick up $P_{\sigma(i+2)}$ (ϵ_1), move to M_1 (δ_1), and load $P_{\sigma(i+2)}$ onto M_1 (ϵ_2). At this point, the MPS cycle with $CRM(\pi_6)$ starts from the state $E = (\Omega, \Omega, \Omega, M_1^-)$.

The waiting time equations corresponding to initialization INA6 are as follows:

$$\begin{aligned} w_1^i(0) &= a_{\sigma(i)}, \\ w_2^i(0) &= \max\{0, b_{\sigma(i)} - 2\delta_1 - 2\delta_2 - \epsilon_1 - \epsilon_2 + \eta\}, \\ w_1^{i+1}(0) &= \max\{0, a_{\sigma(i+1)} - w_2^i(0) - 2\delta_2 - 2\delta_3 - \epsilon_5 - \epsilon_6 + \eta\}. \end{aligned}$$

The waiting time equations corresponding to the qth iteration of the MPS cycle for $CRM(\pi_6)$ are

$$\begin{aligned} w_1^j(q) &= \max\{0, a_{6\sigma(j)} - \max\{\beta_6, b_{6\sigma(j-1)}, c_{6\sigma(j-2)} \\ &\quad - w_1^{j-1}(q-1)\}\}, j = i+2, \\ w_1^j(q) &= \max\{0, a_{6\sigma(j)} - \max\{\beta_6, b_{6\sigma(j-1)}, c_{6\sigma(j-2)} \\ &\quad - w_1^{j-1}(q)\}\}, j = i+3, \ldots, i+n+1, \end{aligned}$$

where $w_3^{j+n}(q) = w_3^j(q), j = 1, \ldots, n$.

(b) Initialization INB6, starting with part $P_{\sigma(i)}$: pick up $P_{\sigma(i)}$ (ϵ_1), move to M_1 (δ_1), load $P_{\sigma(i)}$ onto M_1 (ϵ_2), wait at M_1 ($w_1^i(0) = a_{\sigma(i)}$), unload $P_{\sigma(i)}$ from M_1 (ϵ_3), move to M_2 (δ_2), load $P_{\sigma(i)}$ onto M_2 (ϵ_4), wait at M_2 ($w_2^i(0) = b_{\sigma(i)}$), unload $P_{\sigma(i)}$ from M_2 (ϵ_5), move to M_3 (δ_3), load $P_{\sigma(i)}$ onto M_3 (ϵ_6), move to I ($\delta_1+\delta_2+\delta_3-2\eta$), pick up $P_{\sigma(i+1)}$ (ϵ_1), move to M_1 (δ_1), load $P_{\sigma(i+1)}$ onto M_1 (ϵ_2), wait at M_1 ($w_1^{i+1}(0) = a_{\sigma(i+1)}$), unload $P_{\sigma(i+1)}$ from M_1 (ϵ_3), move to M_2 (δ_2), load $P_{\sigma(i+1)}$ onto M_2 (ϵ_4), move to I ($\delta_1 + \delta_2 - \eta$), pick up $P_{\sigma(i+2)}$ (ϵ_1), move to M_1 (δ_1), and load $P_{\sigma(i+2)}$ onto M_1 (ϵ_2). At this point, the MPS cycle with $CRM(\pi_6)$ starts from the state $E = (\Omega, \Omega, \Omega, M_1^-)$. The waiting time equations corresponding to initialization INB6 are as follows:

$$\begin{aligned} w_1^i(0) &= a_{\sigma(i)}; w_2^i(0) = b_{\sigma(i)}; w_1^{i+1}(0) = a_{\sigma(i+1)}, \\ w_3^i(0) &= \max\{0, c_{\sigma(i)} - a_{\sigma(i+1)} - 4\delta_1 - 4\delta_2 - 2\delta_3 \\ &\quad - 2\epsilon_1 - 2\epsilon_2 - \epsilon_3 - \epsilon_4 + 4\eta\}, \\ w_2^{i+1}(1) &= \max\{0, b_{\sigma(i+1)} - w_3^i(0) - 2\delta_1 - 2\delta_2 - 2\delta_3 - 2\delta_4 \\ &\quad - \epsilon_1 - \epsilon_2 - \epsilon_7 - \epsilon_8 + 3\eta\}, \\ w_1^{i+2}(1) &= \max\{0, a_{\sigma(i+2)} - w_3^i(0) - w_2^{i+1}(1) - 2\delta_2 \\ &\quad - 4\delta_3 - 2\delta_4 - \epsilon_5 - \epsilon_6 - \epsilon_7 - \epsilon_8 + 3\eta\}. \end{aligned}$$

The waiting time equations corresponding to the q^{th} iteration of the MPS cycle for $CRM(\pi_6)$ are

$$\begin{aligned} w_1^j(q) &= \max\{0, a_{6\sigma(j)} - \max\{\beta_6, b_{6\sigma(j-1)}, c_{6\sigma(j-2)} \\ &\quad - w_1^{j-1}(q-1)\}\}, j = i+3, \\ w_1^j(q) &= \max\{0, a_{6\sigma(j)} - \max\{\beta_6, b_{6\sigma(j-1)}, c_{6\sigma(j-2)} \\ &\quad - w_1^{j-1}(q)\}\}, j = i+4, \ldots, i+n+2, \end{aligned}$$

where $w_3^{j+n}(q) = w_3^j(q), j = 1, \ldots, n$. The main result for the convergence of the MPS cycle with $CRM(\pi_6)$ to a steady state follows.

THEOREM 6.8 *Starting from the state E_I and using part schedule σ, the cell in $RF_3|$(free,A,MP,$CRM(\pi_6)$)$|\mu$ will go through at most* $\max\{1, \min_{1\le i\le n}\{s_i\}\}$ *cycles before reaching a steady state, where $s_i = a_{6\sigma(i)} - \max\{\beta_6, b_{6\sigma(i-1)}\}$, $i = 1, \ldots, n$. Furthermore, this bound is attainable.*

Proof. The proof is similar to that of Theorem 6.7. ∎

In this section, we have shown that an RF_3 cell operating under CRM sequences π_1, π_3, π_4, and π_5 reaches a steady state within the production of one MPS, if it starts from an empty cell E_I with the robot at I ready to pick up a part. If the cell starts from E_I and uses CRM sequence π_2 or π_6, then it converges to a steady state in a number of cycles that is bounded by a function of the cell data. Note that for the MPS cycle with $CRM(\pi_2)$ (resp., $CRM(\pi_6)$), the bound derived in Theorem 6.7 (resp., Theorem 6.8) is also valid for the cell starting from any other state. Similarly, it is easy to show that MPS cycles under CRM sequences π_1, π_3, π_4, and π_5 reach a steady state starting from any other state within the production of at most three MPSs.

6.4.3 A Practical Guide to Initializing Robotic Cells

Here we give a practical method for initializing the cell in $RF_3|$(*free,A, MP, $CRM(\pi_2)$*)$|\mu$ (resp., $RF_3|$(*free,A,MP, $CRM(\pi_6)$*)$|\mu$). In the previous subsection, we saw that there are $2n$ possible ways to initialize each of these cells. The following algorithm, Start, selects the initialization that brings the cell to a steady state in the minimum number of MPS cycle iterations. Let h_j^i denote the robot waiting time for part P_i on

machine M_3 (resp., M_1) in the first cycle, using the jth initialization, $i = 1, \ldots, v, j = 1, \ldots, 2n$, where v is the number of parts required to define a cycle in a steady state. The steps shown in italics refer to FindTime2 (resp., FindTime6) in Section 6.4. Let $x^i, i = 1, \ldots, v$, denote the solution found in Step 1, and $z^i, i = 1, \ldots, v$, denote the solution found in Step 2. Note that x^i,$i = 1, \ldots, v$, satisfies the conditions in Lemma 6.11 (resp., Lemma 6.14). Let y^i,$i = 1, \ldots, v$, denote another feasible vector of waiting times that satisfies the conditions of Lemma 6.11 (resp., Lemma 6.14). Given x^i,$i = 1, \ldots, v$, it is easy to determine y^i,$i = 1, \ldots, v$. See Remark 6.8 (resp., Remark 6.10).

Algorithm Start
Input: w_3^i (resp., w_1^i),$LB^i, UB^i, i = 1, \ldots, n$, x^i, y^i, z^i, $i = 1, \ldots, v$.
Step 1: Calculate $h_j^1, j = 1, \ldots, 2n$.
Step 2: If there exists a *Step 1* solution (in FindTime2), then
For $j = 1, \ldots, 2n$, do
If $x^1 \leq h_j^1 \leq y^1$, then $\xi_j = 0$.
If $y^1 < h_j^1$, then $\xi_j = h_j^1 - y^1$.
If $x^1 > h_j^1$, then $\xi_j = x^1 - h_j^1$.
End
End If
Step 3: If there exists a *Step 2* solution (in FindTime2), then
For $j = 1, \ldots, 2n$, do
If $z^1 \geq h_j^1$, then $\xi_j = z^1 - h_j^1$.
If $z^1 < h_j^1$, then $\xi_j = h_j^1 - z^1$.
End
End If
Step 4: Use initialization j, where $j = \text{argmin}_{1 \leq i \leq 2n}\{\xi_j\}$.
Terminate.

It follows from Lemma 6.12 (resp., Lemma 6.15) that either Step 2 or Step 3 of Start is used. In Start, ξ_j measures the difference between the waiting time of the robot for the first part in a given schedule using initialization j and the corresponding steady-state waiting time for that part. We note that since waiting times always increase or decrease by a constant in each cycle when the cell is not in a steady state, the initialization j that minimizes ξ_j brings the cell to a steady state faster than that by other initializations. Algorithm Start, which provides a guide

to initializing the cell, is very useful for practitioners who would like to bring a three-machine cell operating under an $CRM(\pi_2)$ or $CRM(\pi_6)$ to a steady state as fast as possible. This algorithm can be generalized to cells with $m \geq 3$ machines operating under MPS cycles that do not have an E_q^* state at any machine M_q. Note that for MPS cycles that have an E_q^* state, the steady state can be achieved within one cycle, as discussed at the start of Section 6.4 and also by Hall et al. [75].

6.5 Intractable Cycles for Three-Machine Cells

In this section, we show that the recognition version of the optimal part scheduling problems in the MPS cycles corresponding to $CRM(\pi_2)$ and $CRM(\pi_6)$ are strongly NP-complete. Recall that in the other four cycles, the optimal part schedule can be found in polynomial time.

6.5.1 MPS Cycles with the Sequence $CRM(\pi_2)$

We now show that Problem $RF_3|(free,\ A,\ MP,\ CRM(\pi_2))|\mu$ is strongly NP-hard.

THEOREM 6.9 *The recognition version of* $RF_3|(free,A,MP,CRM(\pi_2))|\mu$ *is in the class NP.*

Proof. We need to prove that FindTime2 finds the steady-state value of $T_{2(\sigma)}$ for a given part schedule σ in polynomial time. We consider two cases for the vector of waiting times $w_3^i, i = 1, \ldots, v$, where we let $a_{2\sigma(v+1)} = a_{2\sigma(1)}$ and $a_{2\sigma(v+2)} = a_{2\sigma(2)}$; $b_{2\sigma(v+1)}, b_{2\sigma(v+2)}$, $c_{2\sigma(v+1)}$, and $c_{2\sigma(v+2)}$ are defined similarly.

- Case 1. $w_3^i \leq b_{2\sigma(i+1)} - \beta_2$, $w_3^i \geq b_{2\sigma(i+1)} - a_{2\sigma(i+2)}$, and $w_3^i \leq c_{2\sigma(i+1)} - a_{2\sigma(i+2)} + b_{2\sigma(i+1)} - \beta_2$, $i = 1, \ldots, v$.

  These three constraints define a lower bound LB_1^i and two upper bounds UB_1^i and UB_2^i on w_3^i, $i = 1, \ldots, v$; thus, $\max\{0, LB_1^i\} = LB^i \leq w_3^i \leq UB^i = \min\{UB_1^i, UB_2^i\}$. From Lemma 6.11, if a solution is found in Step 1, then there exists a job j such that $w_3^j = LB^j$ or $w_3^j = UB^j$. Any solution found in Step 1 has cycle time $\frac{v}{n}T_{2(\sigma)} = v\alpha_2 + \sum_{i=1}^{v} a_{2\sigma(i)} + \sum_{i=1}^{v} w_3^i$, where $\sum_{i=1}^{v} w_3^i = \sum_{i=1}^{v}(c_{2\sigma(i)} + b_{2\sigma(i)} - a_{2\sigma(i+1)} - \beta_2)/2$. Thus, $\frac{v}{n}T_{2(\sigma)} = v\alpha_2 + \sum_{i=1}^{v}(a_{2\sigma(i)} + b_{2\sigma(i)} + c_{2\sigma(i)} - \beta_2)/2$. Step 1 finds one possible solution of this type.

- Case 2. Alternatively, at least one of the above constraints is not satisfied for some part $j, 1 \leq j \leq v$. We consider three subcases:

  (a) Case 2A. If $w_3^j > b_{2\sigma(j+1)} - \beta_2$, then $w_3^{j+1} = \max\{0, c_{2\sigma(j+1)} - \max\{\beta_2, a_{2\sigma(j+2)}\}\}$.

  (b) Case 2B. Similarly, if $w_3^j < b_{2\sigma(j+1)} - a_{2\sigma(j+2)}$, then $w_3^{j+1} = \max\{0, c_{2\sigma(j+1)} - \beta_2\}$.

  (c) Case 2C. Similarly, if $w_3^j > c_{2\sigma(j+1)} - a_{2\sigma(j+2)} + b_{2\sigma(j+1)} - \beta_2$, then $w_3^{j+1} = 0$.

Since at least one waiting time value is fixed in Step 2, the solution is unique if it is found there. Step 2 considers all possible solutions of each type.

Lemma 6.12 and that the conditions of Cases 1 and 2 are exhaustive imply that FindTime2 finds the steady-state solution. Steps 1 and 2 each require time $O(n^2)$. ∎

We next show that the problem of finding the best part schedule to be processed in the MPS cycle with $CRM(\pi_2)$ is NP-complete. Consider the following problem, which is known (Garey and Johnson [57]) to be strongly NP-complete.

Numerical Matching with Target Sums (NMTS): Let $\mathcal{X} = \{x_1, \ldots, x_s\}$, $\mathcal{Y} = \{y_1, \ldots, y_s\}$, and $\mathcal{Z} = \{z_1, \ldots, z_s\}$ be sets of positive integers. Does there exist a partition of $\mathcal{Y} \cup \mathcal{Z}$ into disjoint subsets $\tau_1, \ldots, \tau_s$, such that τ_i contains one element y_{j_i} from $\mathcal{Y}$ and one element z_{ℓ_i} from $\mathcal{Z}$, and $x_i = y_{j_i} + z_{\ell_i}$, for $i = 1, \ldots, s$?

As part of this definition, we assume that if a Numerical Matching with Target Sums exists, then $x_j = y_j + z_j, j = 1, \ldots, s$. We further assume without loss of generality that s is even.

Given an arbitrary instance of NMTS, consider the following instance, referred to as Q_1, of the robotic cell scheduling problem: $v = n = 3s, k = 3$, and part set $J = \{J_j^x, 1 \leq j \leq s\} \cup \{J_j^y, 1 \leq j \leq s\} \cup \{J_j^z, 1 \leq j \leq s\}$. Let

$$
\begin{aligned}
a_j^x &= L + x_j - \beta/2, \quad b_j^x = 3L, \quad c_j^x = \beta/2, \quad j = 1, \ldots, s, \\
a_j^y &= 3L - \beta/2, \quad b_j^y = 2L, \quad c_j^y = X - y_j + \beta/2, \quad j = 1, \ldots, s, \\
a_j^z &= 2L - \beta/2, \quad b_j^z = L + X + z_j, \quad c_j^z = \beta/2, \quad j = 1, \ldots, s,
\end{aligned}
$$

where a_j^r (resp., b_j^r, c_j^r) denotes the processing time of part j of type r on machine M_1 (resp., M_2, M_3), $r \in \{x, y, z\}, j = 1, \ldots, s$; $X = \sum_{j=1}^{s} x_j$; $L = 3sX$; $\epsilon_i = \epsilon = X/12, i = 1, \ldots, 8$; $\delta_i = \delta = X/12, i = 1, \ldots, 4$; $\eta = 0$; $\beta = 8\delta + 4\epsilon$. Note that $D = 6sL + sX + Z$ denotes the threshold cycle time, where $Z = \sum_{j=1}^{s} z_j$. We also let $Y = \sum_{j=1}^{s} y_j$, where $X = Y + Z$, and define $\hat{T}_\sigma = \bar{T}_\sigma - \sum_{i=1}^{3s} w_3^i$.

REMARK 6.11 Let Q_1 be an instance of $RF_3|(free,A,MP,CRM(\pi_2))|\mu$ in which $\delta_i = \delta, i = 1, \ldots, 4; \epsilon_i = \epsilon, i = 1, \ldots, 8$; and $\eta = 0$. From Lemma 6.6, the cycle time for the production of n parts under CRM(π_2) for this instance is given by $T_{2(\sigma)} = n\alpha + \bar{T}_\sigma$, where $\alpha = 4\delta + 4\epsilon$,

$$\begin{aligned} \bar{T}_\sigma &= \sum_{i=1}^{n} \max\{\beta, a_{\sigma(i+2)} + \beta/2, b_{\sigma(i+1)} - w_3^i\} + \sum_{i=1}^{n} w_3^i, \\ w_3^i &= \max\{0, c_{\sigma(i)} - \max\{\beta/2, a_{\sigma(i+1)} - \max\{0, b_{\sigma(i)} - \beta - w_3^{i-1}\}\}\}. \end{aligned}$$

LEMMA 6.16 *When the* $3s$ *parts of* Q_1 *are processed in the schedule*

$$\sigma = [J_1^x, J_1^y, J_1^z, J_2^x, J_2^y, J_2^z, \ldots, J_s^x, J_s^y, J_s^z],$$

the robot waiting times at M_3 *are given by* $w_3^i = X - y_q$ *for* $i = 2 + 3(q - 1), 1 \le q \le s$, *and* $w_3^i = 0$ *otherwise. Therefore,* $\sum_{i=1}^{3s} w_3^i = sX - Y$.

Proof. Consider three cases:

- Case 1. $i = 3 + 3(q - 1)$, $1 \le q \le s$. From Remark 6.11, $w_3^i = \max\{0, c_{\sigma(i)} - \max\{\beta/2, a_{\sigma(i+1)} - \max\{0,\ b_{\sigma(i)} - \beta - w_3^{i-1}\}\}\}$. Note that part $J_{\sigma(i)}$ is of type J^z. Substituting the value of $c_{\sigma(i)} = \beta/2$ in the equation for w_3^i, we have

  $$w_3^i = \max\{0, \beta/2 - \max\{\beta/2, a_{\sigma(i+1)} - \max\{0,\ b_{\sigma(i)} - \beta - w_3^{i-1}\}\}\} = 0.$$

- Case 2. $i = 1 + 3(q - 1)$, $1 \le q \le s$. Here, part $J_{\sigma(i)}$ is of type J^x. Substituting the value of $c_{\sigma(i)} = \beta/2$ in the equation for w_3^i, we have

  $$w_3^i = \max\{0, \beta/2 - \max\{\beta/2,\ a_{\sigma(i+1)} - \max\{0,\ b_{\sigma(i)} - \beta - w_3^{i-1}\}\}\} = 0.$$

- Case 3. $i = 2 + 3(q - 1)$, $1 \le q \le s$. From Case 2, we know that $w_3^{i-1} = 0$. Since parts $J_{\sigma(i)}$ and $J_{\sigma(i+1)}$ are, respectively, of type J^y

and J^z, by substituting the values of $a_{\sigma(i+1)}$, $b_{\sigma(i)}$, and $c_{\sigma(i)}$ in the equation for w_3^i, we get

$$\begin{aligned} w_3^i &= \max\{0, X - y_q + \beta/2 - \max\{\beta/2,\ 2L - \beta/2 \\ &\quad - \max\{0, 2L - \beta - w_3^{i-1}\}\}\}, \\ &= \max\{0, X - y_q + \beta/2 - \max\{\beta/2,\ 2L - \beta/2 - 2L + \beta\}\}, \\ &= \max\{0, X - y_q\} = X - y_q. \end{aligned}$$

Thus, $\sum_{i=1}^{3s} w_3^i = sX - Y$. ∎

Note that from Lemma 6.16, $b_{\sigma(i+1)} - w_3^i$ can be expressed as: $b_{\sigma(i+1)} - w_3^i = b_{\sigma(i+1)} - (X - y_q)$ for $i = 2 + 3(q-1), 1 \le q \le s$, and $b_{\sigma(i+1)} - w_3^i = b_{\sigma(i+1)}$ otherwise.

Let $f_{\sigma(i)} = \max\{\beta, b_{\sigma(i)} - w_3^{i-1}\}$ and $e_{\sigma(i)} = a_{\sigma(i)} + \beta/2$. Then, from Remark 6.11, $T_{2(\sigma)} - n\alpha = \bar{T}_\sigma = \sum_{i=1}^{3s} \max\{f_{\sigma(i)}, e_{\sigma(i+1)}\} + sX - Y$. Thus, $\hat{T}_\sigma = \bar{T}_\sigma - \sum_{i=1}^{3s} w_3^i = \sum_{i=1}^{3s} \max\{f_{\sigma(i)}, e_{\sigma(i+1)}\}$. The expression $\sum_{i=1}^{3s} \max\{f_{\sigma(i)}, e_{\sigma(i+1)}\}$ can be interpreted as the cycle time in a two-machine flowshop producing $3s$ jobs, where the processing times of job i on the first machine M_1' and on the second machine M_2' are e_i and f_i, respectively. The exact formulation follows:

- Processing time of J^x type jobs:
  $e_q\ (J_q^x) = L + x_q, 1 \le q \le s$ (on machine M_1'),
  $f_q\ (J_q^x) = 3L - w_3^{i-1}, 1 \le q \le s$ (on machine M_2', where J_q^x is scheduled in the ith position).

- Processing time of J^y type jobs:
  $e_q\ (J_q^y) = 3L, 1 \le q \le s$ (on machine M_1'),
  $f_q\ (J_q^y) = 2L - w_3^{i-1}, 1 \le q \le s$ (on machine M_2', where J_q^y is scheduled in the ith position).

- Processing time of J^z type jobs:
  $e_q\ (J_q^z) = 2L, 1 \le q \le s$ (on machine M_1'),
  $e_q\ (J_q^z) = L + X + z_q - w_3^{i-1},\ 1 \le q \le s$ (on machine M_2', where J_q^z is scheduled in the ith position).

Note that since w_3^i depends on the parts scheduled both before and after the part scheduled in the ith position, this problem has schedule-dependent processing times.

THEOREM 6.10 *The recognition version of* $RF_3|(free,A,MP,CRM(\pi_2))|\mu$ *is strongly NP-complete.*

Proof. The proof is by reduction from Numerical Matching with Target Sums (NMTS). One can easily verify that the entire construction of Q_1 from an arbitrary instance of NMTS is polynomially bounded. From Theorem 6.9, we need only show that there exists a schedule σ for the part set J with cycle time $\bar{T}_\sigma \leq D$ if and only if there exists a solution to NMTS.

($\Rightarrow$) By Lemma 6.16, the schedule σ, shown in Figure 6.4, has cycle time $\hat{T}_\sigma = 6sL + X \Rightarrow \bar{T}_\sigma = 6sL + X + \sum_{i=1}^{3s} w_3^i = 6sL + sX + Z = D$.

M_1'	$L + x_1$	$3L$	$2L$	$\cdots$	$L + x_s$	$3L$	$2L$
M_2'	$L + y_1 + z_1$	$3L$	$2L$	$\cdots$	$L + y_s + z_s$	$3L$	$2L$

Figure 6.4. A Schedule with $\hat{T}_\sigma = 6sL + X$.

($\Leftarrow$) Suppose there exists a schedule σ_0 for J such that $\bar{T}_{\sigma_0}(J) \leq D$. Then we show that σ_0 must take the form of σ. That is, if the first job is chosen without loss of generality to be J_1^x, then the next three jobs have to be of J^y, J^z, and J^x type, respectively, and so on. The basic idea of the proof is to decompose the expression for $\bar{T}_\sigma$ into two components. The first component is $\sum_{i=1}^{3s} w_3^i$. The second component is $\hat{T}_\sigma = \sum_{i=1}^{3s} \max\{f_{\sigma(i)}, e_{\sigma(i+1)}\}$. Then, by finding a lower bound on the first component, we can find the optimal schedule whose contribution to the second component cannot exceed D minus that lower bound. To do so, we prove six facts about σ_0.

Fact 1. In $\sigma_0, 0 \leq \sum_{i=1}^{3s} w_3^i \leq sX - Y$.

It is clear from Remark 6.11 that $w_3^i \leq \max\{0, c_{\sigma(i)} - \beta/2\}$. From the definition of Q_1, $c_{\sigma(i)}$ takes either the value $\beta/2$ or the value $X - y_q + \beta/2 \geq \beta/2$. Thus, $0 \leq w_3^i \leq X - y_q \Rightarrow \sum_{i=1}^{3s} w_3^i \leq sX - Y$.

Fact 2. In σ_0, the idle time on M_1' or M_2' cannot be more than $(sX - Y)$.

The total processing time of jobs on M_1' is $6sL + X$, and since $\bar{T}_{\sigma_0} \leq 6sL + sX + Z$, the total allowable idle time on M_1' is at most $(6sL + sX + Z) - (6sL + X) = sX + Z - X = sX - Y$. Similarly, the total processing time of jobs on M_2' is $(6sL + sX + Z - \sum_{i=1}^{3s} w_3^i)$. Using Fact 1

and $\bar{T}_{\sigma_0} \leq 6sL + sX + Z$, the total allowable idle time on M_2' is at most $(6sL + sX + Z) - (6sL + sX + Z - \sum_{i=1}^{3s} w_3^i) = \sum_{i=1}^{3s} w_3^i \leq sX - Y$.

Fact 3. In σ_0, a J^y type job follows job J_1^x.

If a J^x type job follows job J_1^x, the idle time on M_1' will be at least $(3L - w_3^i) - (L + x_2) = 6sX - x_2 - w_3^i$, as shown in Figure 6.5. Since $6sX - x_2 - w_3^i \geq 4sX + Y > sX - Y$, this contradicts Fact 2. Similarly, if a J^z type job follows job J_1^x, an idle time of $(3L - w_3^i) - 2L = 3sX - w_3^i > sX - Y$ occurs on M_1', as shown in Figure 6.6, which contradicts Fact 2.

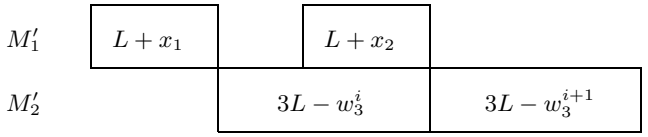


Figure 6.5. Idle Time on M_1' if Job J^x Follows J_1^x.

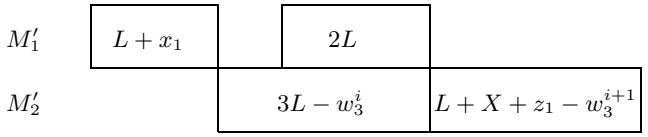


Figure 6.6. Idle Time on M_1' if Job J^z Follows J_1^x.

Fact 4. In σ_0, a J^z type job follows the job schedule $[J_1^x, J_1^y]$.

If a J^x type job follows job J_1^y, the idle time on M_1' will be at least $(2L - w_3^{i+1}) - (L + x_2) = 3sX - x_2 - w_3^{i+1} > sX - Y$, as shown in Figure 6.7, which contradicts Fact 2. Similarly, if a J^y type job follows job J_1^y, an idle time of at least $3L - (2L - w_3^{i+1}) \geq 3sX > sX - Y$ occurs on M_2', as shown in Figure 6.8, which contradicts Fact 2.

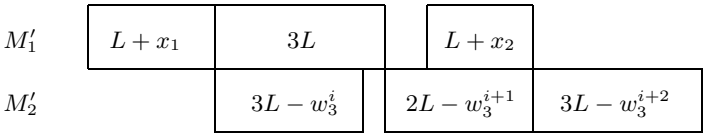


Figure 6.7. Idle Time on M_1' if Job J^x Follows J_1^y.

Fact 5. In σ_0, a J^x type job follows the job schedule $[J_1^x, J_1^y, J_1^z]$.

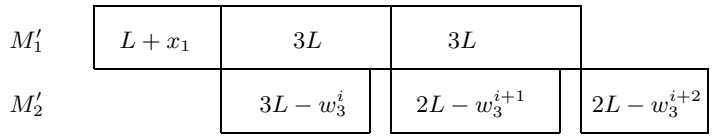


Figure 6.8. Idle Time on M_2' if Job J^y Follows J_1^y.

If a J^y type job follows the job schedule $[J_1^x, J_1^y, J_1^z]$, the idle time on M_2' will be at least $3L - (L + X + z_1 - w_3^{i+2}) \geq 4sX > sX - Y$, as shown in Figure 6.9, which contradicts Fact 2. Similarly, if a J^z type job follows the job schedule $[J_1^x, J_1^y, J_1^z]$, the idle time on M_2' will be at least $2L - (L + X + z_1 - w_3^{i+2}) \geq sX > sX - Y$, as shown in Figure 6.10, which contradicts Fact 2.

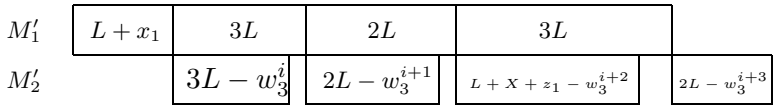


Figure 6.9. Idle Time on M_2' if Job J^y Follows J_1^z.

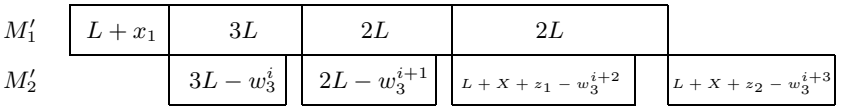


Figure 6.10. Idle Time on M_2' if Job J^z Follows J_1^z.

Fact 6. There is no idle time on machine M_1' in schedule σ_0.

Using an induction argument based on Facts 3, 4, and 5, we can show that the jobs are scheduled in the following order: $[J_1^x, J_1^y, J_1^z, J_2^x, J_2^y, J_2^z, \ldots, J_s^x, J_s^y, J_s^z]$, as shown in Figure 6.4. From Lemma 6.16, we have $\sum_{i=1}^{3s} w_3^i = sX - Y$. Now, $\bar{T}_\sigma = \hat{T}_\sigma + sX - Y \leq 6sL + sX + Z \Rightarrow \hat{T}_\sigma \leq 6sL + X$, which is the total processing time needed on machine M_1'.

If $y_{j_i} + z_{\ell_i} > x_i$ for some $i, 1 \leq i \leq s$, then idle time of length $(y_{j_i} + z_{\ell_i} - x_i) > 0$ occurs on M_1', as shown in Figure 6.11, which contradicts Fact 6. Thus, $y_{j_i} + z_{\ell_i} \leq x_i, i = 1, \ldots, s$, and since $\sum_{i=1}^{s}(y_{j_i} + z_{\ell_i}) = \sum_{i=1}^{s} x_i$, we have $y_{j_i} + z_{\ell_i} = x_i, i = 1, \ldots, s$, which implies the existence of an NMTS. ∎

Figure 6.11. Idle Time on M_1' if $x_i < y_{j_i} + z_{j_i}$.

6.5.2 MPS Cycles with the Sequence $CRM(\pi_6)$

We now show that Problem $RF_3|(free,A,MP,CRM(\pi_6))|\mu$ is strongly NP-hard.

THEOREM 6.11 *The recognition version of* $RF_3|(free,A,MP,CRM(\pi_6))|\mu$ *is in the class NP.*

Proof. We need to prove that FindTime6 finds the steady-state value of $T_{6(\sigma)}$ for a given part schedule σ in polynomial time. We consider two cases for the vector of waiting times $w_1^i, i = 1, \ldots, n$, where we let $a_{6\sigma(n+1)} = a_{6\sigma(1)}, a_{6\sigma(n+2)} = a_{6\sigma(2)}$, and similarly for $b_{6\sigma(n+1)}$, $b_{6\sigma(n+2)}$, $c_{6\sigma(n+1)}$ and $c_{6\sigma(n+2)}$.

- Case 1. $w_1^i \leq c_{6\sigma(i-1)} - \beta_6, w_1^i \leq c_{6\sigma(i-1)} - b_{6\sigma(i)}$, $w_1^i \geq c_{6\sigma(i-1)} - a_{6\sigma(i+1)}$, $i = 1, \ldots, n$.
  These three constraints define a lower bound LB_1^i, and two upper bounds UB_1^i and UB_2^i on w_1^i, $i = 1, \ldots, n$. Thus, $\max\{0, LB_1^i\} = LB^i \leq w_1^i \leq UB^i = \min\{UB_1^i, UB_2^i\}$. From Lemma 6.14, if a solution is found in Step 1, then there exists a job j such that $w_1^j = LB^j$. Any solution found in Step 1 has cycle time $T_{6(\sigma)} = n\alpha_6 + \sum_{i=1}^{n} a_{6\sigma(i)}$. Step 1 finds one possible solution of this type.

- Case 2. Alternatively, at least one of the above constraints is not satisfied for some part $j, 1 \leq j \leq n$. We consider two subcases.

  (a) Case 2A. If $w_1^j > \min\{c_{6\sigma(j-1)} - \beta_6, c_{6\sigma(j-1)} - b_{6\sigma(j)}\}$, then $w_1^{j+1} = \max\{0, a_{6\sigma(j+1)} - \max\{\beta_6, b_{6\sigma(j)}\}\}$.

  (b) Case 2B. Similarly, if $w_1^j < c_{6\sigma(j-1)} - a_{6\sigma(j+1)}$, then $w_1^{j+1} = 0$.

Since at least one waiting time value is fixed in Step 2, the solution is unique if it is found there. Step 2 considers all possible solutions of each type in Cases 2A and 2B.

Lemma 6.15 and that the conditions of Cases 1 and 2 are exhaustive imply that FindTime6 finds the steady-state solution. Steps 1 and 2 each require time $O(n^2)$. ∎

We next show that the problem of finding the best part schedule to be processed in an MPS cycle with $CRM(\pi_6)$ is NP-complete.

Given an arbitrary instance of NMTS, consider the following instance, referred to as Q_2, of the robotic cell scheduling problem $RF_3|(free,A,MP, CRM(\pi_6))|\mu$ with $n = 4s, k = 4$, and part set $J = \{J_j^w|1 \le j \le s\} \cup \{J_j^x|1 \le j \le s\} \cup \{J_j^y|1 \le j \le s\} \cup \{J_j^z|1 \le j \le s\}$. Let

$$\begin{aligned}
a_j^y &= 4L + X, \quad b_j^y = 3L + X, \quad c_j^y = 2L + y_j, \quad j = 1, \ldots, s,\\
a_j^z &= 3L + X, \quad b_j^z = 2L, \quad c_j^z = L + 2X + z_j, \quad j = 1, \ldots, s,\\
a_j^w &= 2L + X, \quad b_j^w = L, \quad c_j^w = 4L + X, \quad j = 1, \ldots, s,\\
a_j^x &= L + X + x_j, \quad b_j^x = 4L + X, \quad c_j^x = 3L + X, \quad j = 1, \ldots, s,
\end{aligned}$$

where a_j^r, b_j^r and c_j^r denote the processing times of part j of type r on machines M_1, M_2, and M_3, respectively, $r \in \{w, x, y, z\}, j = 1, \ldots, s$; $X = \sum_{j=1}^{s} x_j; L = 3sX$. We let $\epsilon_i = \epsilon = X/6, \quad i = 1, \ldots, 8; \delta_i = \delta = X/6, \quad i = 1, \ldots, 4; \eta = 0; D = 10sL + 4sX + X$.

Remark 6.12 Q_2 is an instance of $RF_3|(free,A,MP,CRM(\pi_6))|\mu$ with $\delta_i = \delta, \epsilon_i = \epsilon$, and $\eta = 0$. From Lemma 6.7, the cycle time for the production of n parts under CRM π_6 for this instance is given by $T_{6(\sigma)} = n\alpha + \bar{T}_\sigma$, where $\alpha = 4\delta + 4\epsilon$, $\beta = 8\delta + 4\epsilon$,

$$\begin{aligned}
\bar{T}_\sigma &= \sum_{i=1}^{n} \max\{\beta, c_{\sigma(i)} - w_1^{i+1}, b_{\sigma(i+1)}, a_{\sigma(i+2)}\},\\
w_1^{i+1} &= \max\{0, a_{\sigma(i+1)} - \max\{\beta, b_{\sigma(i)}, c_{\sigma(i-1)} - w_1^i\}\}.
\end{aligned}$$

Lemma 6.17 *When the $4s$ parts of Q_2 are processed in the schedule*

$$\sigma = [J_1^y, J_1^z, J_1^w, J_1^x, J_2^y, J_2^z, J_2^w, J_2^x, \ldots, J_s^y, J_s^z, J_s^w, J_s^x],$$

the robot waiting times at M_1 are given by
$w_1^i = X - y_j, i = 3 + 4(j-1), 1 \le j \le s$, *and*
$w_1^i = 0$ *otherwise.*

Proof. Consider four cases.

- Case 1. $i = 1 + 4(j-1), 1 \le j \le s$.
  From Remark 6.12, $w_1^i = \max\{0, a_{\sigma(i)} - \max\{\beta, b_{\sigma(i-1)}, c_{\sigma(i-2)} - w_1^{i-1}\}\}$, where $J_{\sigma(i)}$ is of type J^y, $J_{\sigma(i-1)}$ is of type J^x, and $J_{\sigma(i-2)}$ is of type J^w. Thus, $w_1^i = \max\{0, 4L + X - \max\{2X, 4L + X, 4L + X - w_1^{i-1}\}\} = 0$.

- Case 2. $i = 2 + 4(j-1), 1 \le j \le s$.
  Here $J_{\sigma(i)}$ is of type J^z, $J_{\sigma(i-1)}$ is of type J^y, and $J_{\sigma(i-2)}$ is of type J^x. Using the fact that $w_1^{i-1} = 0$ from Case 1, we have $w_1^i = \max\{0, 3L + X - \max\{2X, 3L + X, 3L + X\}\} = 0$.

- Case 3. $i = 3 + 4(j-1), 1 \le j \le s$.
  In this case, $J_{\sigma(i)}$ is of type J^w, $J_{\sigma(i-1)}$ is of type J^z, and $J_{\sigma(i-2)}$ is of type J^y. Since $w_1^{i-1} = 0$ from Case 2, we have $w_1^i = \max\{0, a_{\sigma(i)} - \max\{\beta, b_{\sigma(i-1)}, c_{\sigma(i-2)}\}\} = \max\{0, 2L + X - \max\{2X, 2L, 2L + y_j\}\} = \max\{0, 2L + X - (2L + y_j)\} = X - y_j$.

- Case 4. $i = 4 + 4(j-1), 1 \le j \le s$.
  Here $J_{\sigma(i)}$ is of type J^x, $J_{\sigma(i-1)}$ is of type J^w, and $J_{\sigma(i-2)}$ is of type J^z. From Case 3, $w_1^{i-1} = X - y_j$; therefore, $w_1^i = \max\{0, L + X + x_j - \max\{2X, L, L + 2X + z_j - (X - y_j\}\} = \max\{0, L + X + x_j - (L + X + z_j + y_j)\} = \max\{0, x_j - y_j - z_j\} = 0$. ∎

THEOREM 6.12 *The recognition version of $RF_3|(free,A,MP,CRM(\pi_6))|\mu$ is strongly NP-complete.*

Proof. It is easy to see that the construction of Q_2 from an arbitrary instance of NMTS is polynomially bounded. From Theorem 6.11, we need only show that there exists a schedule σ for the part set J with $\bar{T}_\sigma \le D$ if and only if there exists a solution to NMTS.
($\Rightarrow$) We may think of $\bar{T}_\sigma$ as the cycle time in a flowshop with three machines (M_1', M_2', M_3') and schedule-dependent processing times. In this flowshop the processing time of job $\sigma(i)$ on M_3' depends on the waiting time w_1^{i+1}, which in turn depends on w_1^i and the processing times of jobs $J_{\sigma(i+1)}, J_{\sigma(i)}$, and $J_{\sigma(i-1)}$ on M_1', M_2' and M_3', respectively. Using Lemma 6.17, the schedule σ shown in Figure 6.12 has $\bar{T}_\sigma = 10sL + 4sX + X = D$.
($\Leftarrow$) Suppose there exists a schedule σ_0 such that $\bar{T}_{\sigma_0}(J) \le D$. We first prove that σ_0 must take the form of σ. Since the schedule is cyclic, we

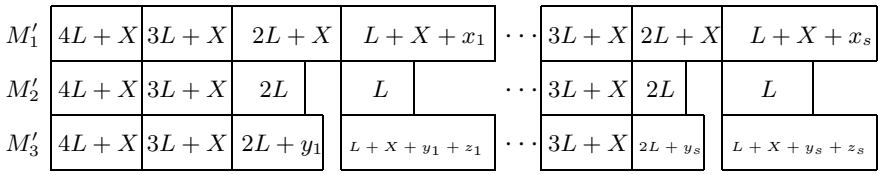


Figure 6.12. A Schedule with $\bar{T}_\sigma = D$.

may assume without loss of generality that the first job in σ_0 is J_1^y. We then prove that the next four jobs in σ_0 must be of types J^z, J^w, J^x and J^y, in that order; prove the same for the next four jobs; and so on until all $4s$ jobs in J have been scheduled. To do so, we prove five facts about σ_0.

Fact 1. There is no idle time on machine M_1' in schedule σ_0.

This follows since the total job processing time on M_1' equals D.

Fact 2. The total idle time on machine M_2' is at most $(2s+1)X$.

This follows since the total job processing time on M_2' equals $10sL + 2sX = D - (2s+1)X$.

Fact 3. In σ_0, a J^z type job follows job J_1^y.

If a J^y type job follows job J_1^y, then the idle time on M_2' will be at least $L = 3sX > (2s+1)X$ (see Figure 6.13), which contradicts Fact 2.

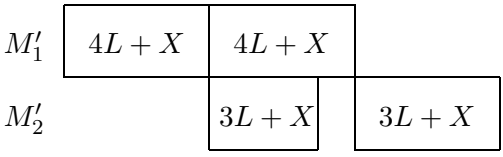


Figure 6.13. Idle Time on M_2' if Job J^y Follows J_1^y.

Alternatively, if a J^w (resp., J^x) type job follows job J_1^y, then an idle time is created on M_1', as shown in Figure 6.14 (resp., 6.15), which contradicts Fact 1.

Fact 4. In σ_0, a J^w type job follows the job schedule $[J_1^y, J_1^z]$.

If a J^y (resp., J^z) type job follows job J_1^z, then the idle time on M_2' will be at least $2L + X > (2s+1)X$ (resp., at least $L + X > (2s+1)X$), as shown in Figure 6.16 (resp., Figure 6.17), which contradicts Fact 2.

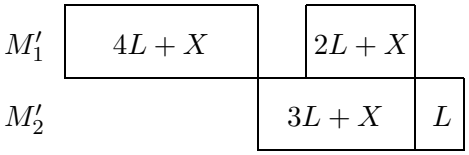


Figure 6.14. Idle Time on M_1' if Job J^w Follows J_1^y.

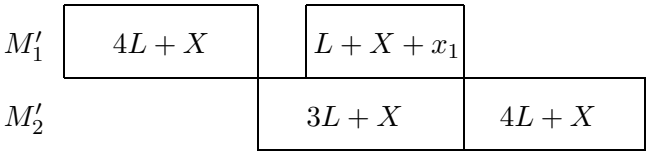


Figure 6.15. Idle Time on M_1' if Job J^x Follows J_1^y.

On the other hand, if a J^x type job follows J_1^z, an idle time is created on Machine M_1', as shown in Figure 6.18, which contradicts Fact 1.

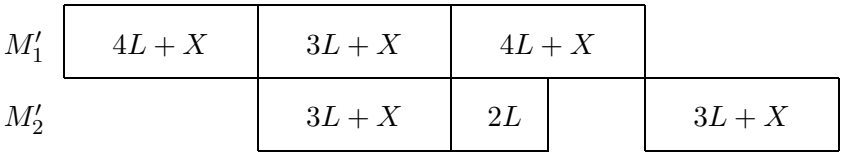


Figure 6.16. Idle Time on M_2' if Job J^y Follows J_1^z.

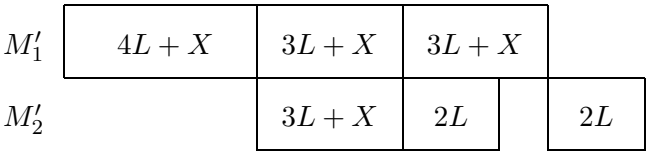


Figure 6.17. Idle Time on M_2' if Job J^z Follows J_1^z.

Fact 5. In σ_0, the schedule $[J_1^y, J_1^z, J_1^w]$ is followed by a J^x type job, and then by a J^y type job.

If any job except a J^x type job follows job J_1^w, then the total idle time on machine M_2' will exceed $L > (2s + 1)X$, as shown in Figures 6.19, 6.20, and 6.21, which contradicts Fact 2. Similarly, if any job except a J^y type job follows the schedule $[J_1^y, J_1^z, J_1^w, J_1^x]$, an idle time of at least

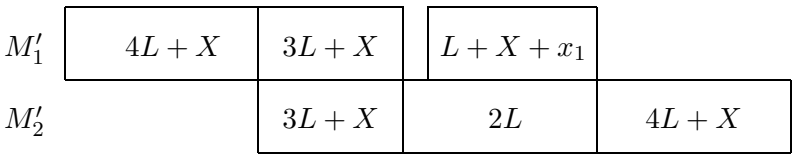


Figure 6.18. Idle Time on M_1' if Job J^x Follows J_1^z.

$L > 0$ is created on machine M_1', as shown in Figures 6.22, 6.23, and 6.24, which contradicts Fact 1.

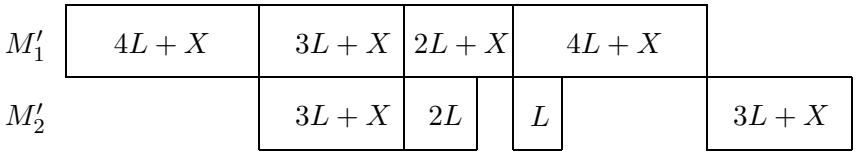


Figure 6.19. Idle Time on M_2' if Job J^y Follows J_1^w.

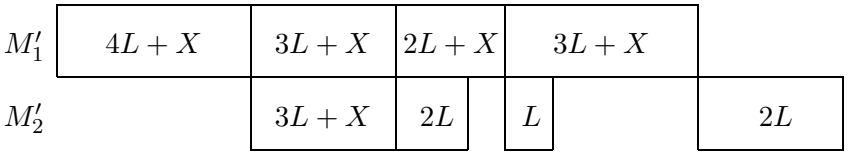


Figure 6.20. Idle Time on M_2' if Job J^z Follows J_1^w.

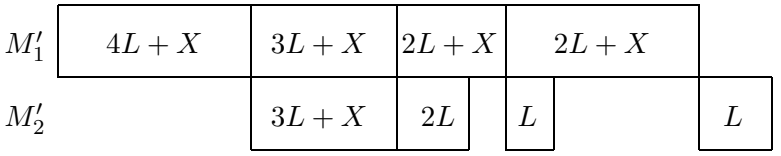


Figure 6.21. Idle Time on M_2' if Job J^w Follows J_1^w.

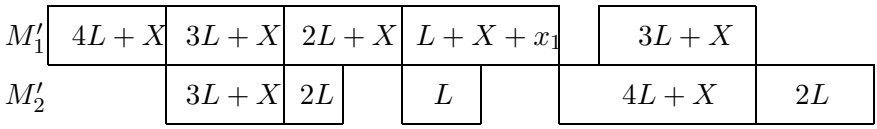


Figure 6.22. Idle Time on M_1' if Job J^z Follows J_1^x.

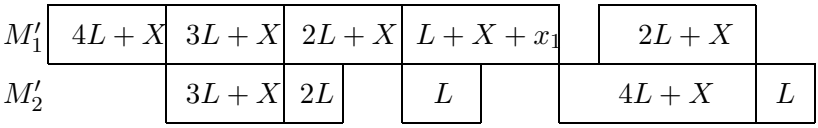


Figure 6.23. Idle Time on M_1' if Job J^w Follows J_1^x.

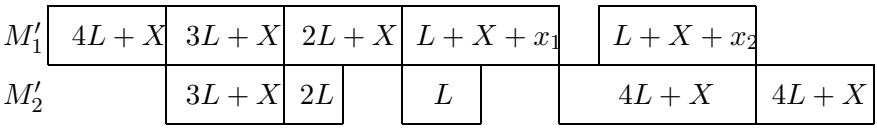


Figure 6.24. Idle Time on M_1' if Job J^x Follows J_1^x.

An induction argument using Facts 1 to 5 can be used to show that schedule σ_0 has the form of schedule σ. Finally, if $y_{j_i} + z_{\ell_i} > x_i$ for some $i, 1 \leq i \leq s$, then an idle time of length $(y_{j_i} + z_{\ell_i} - x_i) > 0$ is created on machine M_1', as shown in Figure 6.25, which contradicts Fact 1.

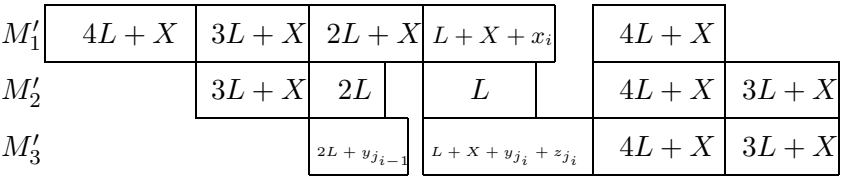


Figure 6.25. Idle Time on M_1' if $x_i < y_{j_i} + z_{j_i}$.

Thus, $y_{j_i} + z_{\ell_i} \leq x_i, i = 1, \ldots, s$. Furthermore, since $\sum_{i=1}^{s}(y_{j_i} + z_{\ell_i}) = \sum_{i=1}^{s} x_i$, we have $y_{j_i} + z_{\ell_i} = x_i, i = 1, \ldots, s$, which implies the existence of an NMTS. ∎

6.5.3 Complexity of Three-Machine Robotic Cells

We show that finding an optimal MPS cycle among the cycles defined using the CRM sequences $\pi_1, \ldots, \pi_6$ is strongly NP-hard. Here we use

the expressions for the cycle time developed for the CRM sequences $\pi_1, \ldots, \pi_6$ in Section 6.3.1. We study the problem of minimizing cycle time without restricting to any specific CRM sequence or even to the class of CRM sequences. We show that the problem is strongly NP-hard in either case.

THEOREM 6.13 *The recognition version of $RF_3|(free,A,MP,CRM)|\mu$ is strongly NP-complete.*

Proof. From Theorem 6.12, it suffices to show that the MPS cycle with $CRM(\pi_6)$ is the only cycle that can give $\bar{T}_\sigma \leq D = 10sL + 4sX + X$ for instance Q_2. For notational convenience, we let the processing times of the $4s$ parts in Q_2 be denoted as $a_i, b_i, c_i, 1 \leq i \leq 4s$, on machines M_1', M_2', and M_3', respectively. Then $D = \sum_{i=1}^{4s} a_i = 10sL + 4sX + X$. Let $a_{\sigma(4s+1)} = a_{\sigma(1)}, a_{\sigma(4s+2)} = a_{\sigma(2)}$, and similarly for $b_{\sigma(4s+1)}, b_{\sigma(4s+2)}, c_{\sigma(4s+1)}$, and $c_{\sigma(4s+2)}$. We make the following observations.

1. From Theorem 6.2, $\bar{T}_{1(\sigma)} = \sum_{i=1}^{4s}(a_i + b_i + c_i) > 30sL > D$.
2. From Remark 6.11, $\bar{T}_{2(\sigma)} = \sum_{i=1}^{4s} \max\{\beta, a_{\sigma(i+2)} + \beta/2, b_{\sigma(i+1)} - w_3^i\}$, where $w_3^i \geq 0 \Rightarrow \bar{T}_{2(\sigma)} \geq \sum_{i=1}^{4s} a_{\sigma(i+2)} + 2s\beta > D$.
3. From Theorem 6.3, $\bar{T}_{3(\sigma)} \geq \sum_{i=1}^{4s} \max\{\beta_3 + a_{\sigma(i)}, b_{3\sigma(i)} + a_{\sigma(i)}, c_{3\sigma(i)}\} \geq 4s\beta_3 + \sum_{i=1}^{4s} a_{\sigma(i)} > D$.
4. From Theorem 6.4, $\bar{T}_{4(\sigma)} \geq \sum_{i=1}^{4s} b_{\sigma(i)} + \sum_{i=1}^{4s} \max\{\beta_4, a_{4\sigma(i)}, c_{4\sigma(i)}\} > \sum_{i=1}^{4s} a_{4\sigma(i)} > \sum_{i=1}^{4s} a_{\sigma(i)} = D$.
5. From Theorem 6.5, $\bar{T}_{5(\sigma)} \geq \sum_{i=1}^{4s} \max\{a_{5\sigma(i+1)}, \beta_5 + c_{\sigma(i)}, b_{5\sigma(i)} + c_{\sigma(i)}\} \geq \sum_{i=1}^{4s} a_{5\sigma(i)} > \sum_{i=1}^{4s} a_{\sigma(i)} = D$.

■

To conclude this section, we consider a more general problem in which an MPS cycle is not restricted to be a cycle formed by a CRM sequence. Thus, combinations of sequences $\pi_1, \ldots, \pi_6$, are permitted.

THEOREM 6.14 *The recognition version of $RF_3|(free,A,MP,cyclic\text{-}n)|\mu$ is strongly NP-complete.*

Proof. As shown in Figure 6.12, the MPS cycle with $CRM(\pi_6)$ achieves the optimal cycle time of $4s\alpha + D$ for problem instance Q_2. Since

$\sum_{i=1}^{4n} a_i = D$ in instance Q_2, whenever a part finishes processing on machine M_1', it must be replaced by another as soon as possible, i.e., in α time. Equivalently, the robot must perform the following operations with times shown: unload (ϵ), move to M_2' (δ), load (ϵ), move to I (2δ), pick up a new part (ϵ), move to M_1' (δ), and load (ϵ). This sequence of moves takes time $\alpha = 4\delta + 4\epsilon = 4X/3$. We now show that the use of any multi-unit robot move cycle will create a larger cycle time. The multi-unit cycles can be formed by a combination of sequences $\pi_1, \ldots, \pi_6$. More precisely, we show that any combination other than π_6 will increase the cycle time for the problem instance Q_2.

Under π_1, the robot waits at M_2' and M_3' before returning to M_1', which takes time more than $3L > 4X/3$.

Under π_2, the robot unloads (ϵ), moves to M_2' (δ), loads (ϵ), moves to M_3' (δ), unloads (ϵ), moves to O (δ), drops the part (ϵ), moves to I (4δ), picks up a new part (ϵ), moves to M_1' (δ), and loads (ϵ), for a total time of $8\delta + 6\epsilon = 14X/6 > 4X/3$.

Under π_3, the robot unloads (ϵ), moves to M_2' (δ), loads (ϵ), moves to M_3' (δ), unloads (ϵ), moves to O (δ), drops the part (ϵ), moves to M_2' (2δ), unloads (ϵ), moves to M_3' (δ), loads (ϵ), moves to I (3δ), picks up a new part (ϵ), moves to M_1' (δ), and loads (ϵ), for a total time of $10\delta + 8\epsilon = 18X/6 > 4X/3$ time.

Under π_4, the robot waits at M_2' before returning to M_1', which takes at least L units of time with $L > 4X/3$.

Let $J_o^{'}$ and J_o denote the jobs that are processed concurrently on machines $M_1^{'}$ and $M_3^{'}$, respectively, in an optimal schedule. Then ($J_o^{'}$, J_o) must be one of the following pairs of jobs: (J^y, J^w), (J^z, J^x),(J^w, J^y) or (J^x, J^z), as in Figure 6.12. This is because if any other pair of jobs is processed concurrently on machines $M_1^{'}$ and $M_3^{'}$, then $T_\sigma^{'} > D$.

Under π_5, immediately after loading a part on M_1', the robot moves to M_2' (δ), unloads a part (ϵ) (let the part be J_o), moves to M_3' (δ), loads (ϵ), waits at M_3' (c_o = processing time of J_o on M_3'), unloads (ϵ), moves to O (δ), drops the part (ϵ), moves to M_1' (3δ), for a total time of $c_o + 6\delta + 4\epsilon$. Note that in an optimal schedule, whenever a part finishes processing on machine $M_1^{'}$, it must be removed immediately. Thus, the following inequality must hold: $c_o + 6\delta + 4\epsilon \leq c_o^{'}$, where $c_o^{'}$ is the processing time of job $J_o^{'}$ on $M_1^{'}$. We now show that this inequality does

not hold in the four cases listed above.
Case 1. $(J_o^{'}, J_o) = (J^y, J^w)$. Here $c_o^{'} = 4L + X$, $c_o = 4L + X \Rightarrow c_o + 6\delta + 4\epsilon > c_o^{'}$.
Case 2. $(J_o^{'}, J_o) = (J^z, J^x)$. Here $c_o^{'} = 3L + X$, $c_o = 3L + X \Rightarrow c_o + 6\delta + 4\epsilon > c_o^{'}$.
Case 3. $(J_o^{'}, J_o) = (J^w, J^y)$. Here $c_o^{'} = 2L + X$, $c_o = 2L + y_i \Rightarrow c_o + 6\delta + 4\epsilon > c_o^{'}$, since $\delta = \epsilon = X/6$.
Case 4. $(J_o^{'}, J_o) = (J^x, J^z)$. Here $c_o^{'} = L + X + x_i$, $c_o = L + 2X + z_j \Rightarrow c_o + 6\delta + 4\epsilon > c_o^{'}$.

We have therefore shown that any use of MPS cycles with CRM sequences $\pi_1, \ldots, \pi_5$ alone or in combination with π_6 implies that $T'_\sigma > D$. Thus, the MPS cycle with $CRM(\pi_6)$ is optimal for instance Q_2, and the result follows from Theorem 6.12. ∎

6.6 Scheduling Multiple Part-Types in Large Cells

Because of the interdependence of the robot's waiting times at the different machines, the expression for the cycle time is often recursive. The complexity of the part scheduling problem $RF_m|(free, A,\ MP, CRM)|\mu$ typically increases with the number of non-zero partial waiting times in the cycle. Following this observation, Sriskandarajah et al. [147] develop criteria to assess the complexity of the part scheduling problem in larger ($m \geq 4$) cells. Each MPS cycle is placed into one of four classes contingent on these criteria, which are based on the amount and the locations of the robot's partial waiting in the 1-unit sequence on which the CRM sequence is predicated, and whether the cell reaches a state $\bar{E}_q$ for some $q \in \{2, \ldots, m\}$; $\bar{E}_q$ is the state in which all machines except M_q are free, the robot has just completed loading a part onto M_q, and the robot is about to move to the input hopper I to perform activity A_0 (compare this to the definition of state E_q^* in Section 6.4). Since we are dealing with larger cells, we now denote the processing time of part i on machine M_j by $p_{i,j}$.

Because we are only considering CRM sequences, if an MPS cycle reaches state $\bar{E}_q$ for some q, then this will be the state of the cell each time a new part enters the cell. Therefore, there are never more than two

parts in the cell at any given time. This implies that the time elapsed during the interval between any two consecutive occurrences of $\bar{E}_q$ can be expressed in terms of the machines' processing times on the two parts in the cell during this interval and a constant that represents the time for the robot's actions (movement, load/unload). Hence, the problem can be analyzed as a traveling salesman problem (TSP): each part is a city, and the distance between a pair of cities is the duration of the interval between the corresponding consecutive occurrences of $\bar{E}_q$. Specifically, the distance between cities j and k is the time of the interval between the loading of part j onto M_q and the loading of part k onto M_q. Thus, in this case, finding an optimal MPS cycle is equivalent to finding an optimal TSP tour.

Recall that a CRM sequence $CRM(\pi_{g,m})$ is a robot move sequence in an m-machine cell in which a 1-unit robot move sequence $\pi_{g,m}$ is repeated exactly n times in succession, where n is the number of parts to be produced in an MPS cycle. Also, not all 1-unit sequences have the same number of partial waiting times. For example, a 1-unit robot move sequence in which the robot loads and then stays with a part during processing at each machine will have no partial waiting times, but all other 1-unit robot move sequences will have at least one partial waiting time. Let $\mathcal{N}$ denote the total number of partial waiting times in a 1-unit robot move sequence. We let $\mathcal{N}^i$ (respectively, $\mathcal{N}_j$) denote the number of partial waiting times on machines $M_1, \ldots, M_i$ (resp., $M_j, \ldots, M_m$) in a given robot move sequence, where $i \leq m$ and $j \geq 1$. It follows that $\mathcal{N} = \mathcal{N}^m = \mathcal{N}_1$ denotes the total number of partial waiting times in a robot move sequence.

To prove some of the results of this section requires using an alternative notation for robot movements. The unloading of machine M_i is designated by M_i^+, and the loading of machine M_i is designated by M_i^-. One can think of M_i^+ as indicating that a part is added to the robot, whereas M_i^- indicates that a part is subtracted from the robot. Cycles in an m-machine cell are typically specified by a collection of M_i^- symbols, with M_m^+ at the beginning and at the end. For example, $\pi_{6,3} = \{M_3^+, M_3^-, M_2^-, M_1^-, M_3^+\}$. 1-unit cycles are defined in Chapter 3 and Appendix A.

Sriskandarajah et al. [147] classify the MPS cycles defined using CRM sequences in an m-machine cell, for $m \geq 2$, according to the tractability

of their associated part scheduling problems. More specifically, MPS cycles formed using sequences $CRM(\pi_{g,m})$ are classified as follows:

- **Class** U: schedule independent (trivially solvable).
- **Class** V: TSP models.
- **Class** W: strongly NP-hard, but do not have a natural TSP structure (in the sense defined later in Section 6.6.3).

Class V can be further divided into two subclasses:

- **Class** $V1$: TSP special cases (solvable in polynomial time using the Gilmore-Gomory algorithm)
- **Class** $V2$: strongly NP-hard TSP models.

Sections 6.6.1–6.6.4 examine these classes in turn. The main result of our analysis is Theorem 6.15 (Section 6.6.5). Before we begin our analysis, we define several easily verifiable conditions (C1–C3) on a CRM sequence. As we will prove, these conditions determine the complexity of the part scheduling problem associated with a CRM sequence.
C1: $\mathcal{N} = 0$.
C2a: $\mathcal{N} = 2$.
C2b: A partial waiting occurs either at machine M_1 or at M_m, or both.
C3: The cycle reaches state $\bar{E}_q$ at machine M_q, where $1 < q \leq m$.

The following discussion is needed to clarify the relationships between Classes $U, V1, V2$, and W and Conditions C1–C3. Clearly, Condition C1 is incompatible with either Condition C2a or Condition C2b. Since the definition of $\bar{E}_q$ implies that $\mathcal{N} \geq 1$, Condition C1 is incompatible with Condition C3. Finally, as will be shown in Corollary 6.1, Condition C2a implies Condition C3. These relationships are illustrated in Table 6.2, where infeasible combinations of conditions are omitted. Note that the three (resp., two) columns for Class $V2$ (resp., W) represent the different conditions under which this class of cycles occurs. As will be discussed in the proof of Lemma 6.29, there are no robot move sequences in which $\mathcal{N} = 1$.

It should be noted that MPS cycles formed by $\pi_{1,2}$ and $\pi_{1,3}$ are U-cycles with $\mathcal{N} = 0$. Cycles formed using $\pi_{2,2}$, $\pi_{3,3}$, $\pi_{4,3}$ and $\pi_{5,3}$ are $V1$-cycles, since they contain an $\bar{E}_q$ state, $\mathcal{N} = 2$, and one of the partial waiting occurs either on the first machine or on the last machine. However,

the cycles formed by $CRM(\pi_{2,3})$ and $CRM(\pi_{6,3})$ are W-cycles, since they do not contain an $\bar{E}_q$ state and $\mathcal{N} > 2$. The MPS cycles formed by the following 1-unit move sequences are $V2$-cycles in a four-machine cell: $\pi_{9,4} = (A_0, A_1, A_3, A_4, A_2) = (M_4^+, M_3^-, M_1^-, M_2^-, M_4^-, M_4^+)$ and $\pi_{10,4} = (A_0, A_3, A_1, A_4, A_2) = (M_4^+, M_3^-, M_1^-, M_4^-, M_2^-, M_4^+)$. The reader may verify the following property for V-cycles: At most two parts may be present in the cell at any given time. This property leads to tractable TSP models for these cycles. The reader may also verify the following property for W-cycles: At least three or more parts may be present in the cell at some point in time. This property makes the problem more difficult, and leads to a strongly NP-hard classification.

Class	U	$V1$	$V2$	$V2$	$V2$	W	W
C1	Yes	No	No	No	No	No	No
C2a	No	Yes	Yes	No	No	No	No
C2b	No	Yes	No	Yes	No	Yes	No
C3	No	Yes	Yes	Yes	Yes	No	No

Table 6.2. Classes of MPS Cycles with CRM Sequences.

6.6.1 Class U: Schedule Independent Problems

We begin with a preliminary result.

LEMMA 6.18 *In any CRM sequence that satisfies Condition C1 in an m-machine cell, at most one part is in the cell at any point in time.*

Proof. If $\mathcal{N} = 0$, then when the robot is about to pick up a part $P_{\sigma(j)}$ at I, the cell must be in state E_I, where all machines are free. Since the robot move sequence repeats after one unit, there must also be a point in time at which all machines are free and the robot is about to pick up part $P_{\sigma(j+1)}$ at I. Thus, part $P_{\sigma(j)}$ must leave the cell before part $P_{\sigma(j+1)}$ enters. ∎

This leads to the following result.

LEMMA 6.19 *The optimal part schedule in an m-machine cell can be found trivially in a CRM sequence that satisfies Condition C1.*

Proof. It follows from Lemma 6.18 that in a CRM sequence with $\mathcal{N} = 0$, the cycle time can be written as the sum of the processing times of the

parts plus the sum of movement times of the robot. Since both are independent of the part schedule, all part schedules provide the same cycle time. It also follows from Lemma 6.18 that there is exactly one U-cycle in an m-machine cell, since the robot waits at every machine while a part is being processed. ∎

The U-cycle is an MPS cycle formed by $CRM(\pi_{1,m})$, where $\pi_{1,m}=(A_0, A_1, A_2, \ldots, A_m) = \{M_m^+, M_1^-, M_2^-, \ldots, M_m^-, M_m^+\}$. Notations for 1-unit cycles are given in Chapter 3.

6.6.2 Class $V1$: Special Cases of the TSP

We show that if a robot move sequence satisfies Conditions C2a and C2b, then the associated part scheduling problem can be formulated as a special case of the TSP, and solved using the Gilmore-Gomory algorithm. The MPS cycles formed by robot move sequences that satisfy these conditions are called $V1$-cycles.

LEMMA 6.20 *The optimal part schedule in an m-machine cell can be found in time $O(n \log n)$ in the $(2m-3)$ MPS cycles that satisfy Conditions C2a and C2b.*

Proof. In MPS cycles, the associated CRM sequences satisfying the lemma's conditions have waiting times at the following machines: $(M_1, M_2), \ldots, (M_1, M_m), (M_2, M_m), \ldots, (M_{m-1}, M_m)$. Thus, there are $2m-3$ such MPS cycles.

For each of the two cases below, the cycle time for $CRM(\pi_{j,m})$ can be written as $T_{j(\sigma)} = \sum_{i=1}^{n} T^h_{j\sigma(i)\sigma(i+1)}$, where $\sum_{i=1}^{n} T^h_{j\sigma(i)\sigma(i+1)} = Z + \sum_{i=1}^{n} \max\{e_{\sigma(i+1)}, f_{\sigma(i)}\}$ for some constant Z, and where we let $\sigma(n+1) = \sigma(1)$. The two parameters $e_{\sigma(i)}$ and $f_{\sigma(i)}$ can be associated with city i in the TSP, where the distance between cities i and $i+1$ is $\max\{e_{\sigma(i+1)}, f_{\sigma(i)}\}$. An instance of the TSP with this distance matrix can be solved by the Gilmore-Gomory algorithm (Remark 6.1 and Appendix B).

Case 1. There is a waiting time at machine M_1 and at M_s for some $2 \le s \le m$.

If there is not an $\bar{E}_s$ state at machine M_s, then the robot must move to M_1 after loading M_s. In this case, since $\mathcal{N} = 2$, two parts will eventually be on M_s simultaneously, which is not possible. Thus, the MPS cycle reaches an E_s state at M_s. Starting from that state with part $P_{\sigma(i)}$ at M_s,

cycle π_h includes the following activities: move to I ($\sum_{q=1}^{s}\delta_q-(s-1)\eta$), pick up a new part $P_{\sigma(i+1)}$ (ϵ_1), move to M_1 (δ_1), load $P_{\sigma(i+1)}$ onto M_1 (ϵ_2), move from M_1 to M_s ($\sum_{q=2}^{s}\delta_q-(s-2)\eta$), wait (if necessary) at M_s (w_s^i), unload $P_{\sigma(i)}$ from M_s (ϵ_{2s+1}), move to M_{s+1} (δ_{s+1}), load $P_{\sigma(i)}$ onto M_{s+1} (ϵ_{2s+2}), wait at M_{s+1} ($p_{\sigma(i),s+1}$), unload (ϵ_{2s+3}), ..., drop $P_{\sigma(i)}$ at O (ϵ_{2m+2}), move to M_1 ($\sum_{q=2}^{m+1}\delta_q-(m-1)\eta$), wait (if necessary) at M_1 (w_1^{i+1}), unload $P_{\sigma(i+1)}$ from M_1 (ϵ_3), move to M_2 (δ_2), load $P_{\sigma(i+1)}$ onto M_2 (ϵ_4), wait at M_2 ($p_{\sigma(i+1),2}$), ..., and load $P_{\sigma(i+1)}$ on M_s (ϵ_{2s}). Thus,

$$\begin{aligned}
T^s_{h\sigma(i)\sigma(i+1)} &= \sum_{q=1}^{2m+2}\epsilon_q+2\sum_{q=1}^{m+1}\delta_q+2\sum_{q=2}^{s}\delta_q-m\eta-2s\eta+4\eta \\
&\quad+\sum_{t=2}^{s-1}p_{\sigma(i+1),t}+\sum_{t=s+1}^{m}p_{\sigma(i),t}+w_1^{i+1}+w_s^i, \text{ where} \\
w_1^{i+1} &= \max\{0,p_{\sigma(i+1),1}-2\sum_{q=2}^{m+1}\delta_q+(m+s-3)\eta-\sum_{q=2s+1}^{2m+2}\epsilon_q \\
&\quad-\sum_{t=s+1}^{m}p_{\sigma(i),t}-w_s^i\}, \text{ and} \\
w_s^i &= \max\{0,p_{\sigma(i),s}-2\sum_{q=1}^{s}\delta_q+(2s-3)\eta-\epsilon_1-\epsilon_2\}.
\end{aligned}$$

Substituting for w_1^{i+1}, we obtain
$T^s_{h\sigma(i)\sigma(i+1)}=\sum_{q=1}^{2s}\epsilon_q+2\sum_{q=1}^{s}\delta_q+2(s-1)\delta_0-s\eta+\eta+\sum_{t=2}^{s-1}p_{\sigma(i+1),t}+\max\{2\sum_{q=2}^{m+1}\delta_q-(m+s-3)\eta+\sum_{q=2s+1}^{2m+2}\epsilon_q+\sum_{t=s+1}^{m}p_{\sigma(i),t}+w_s^i, p_{\sigma(i+1),1}\}$. Letting $e_{\sigma(i+1)}=p_{\sigma(i+1),1}$ and $f_{\sigma(i)}=2\sum_{q=2}^{m+1}\delta_q-(m+s-3)\eta+\sum_{q=2s+1}^{2m+2}\epsilon_q+\sum_{t=s+1}^{m}p_{\sigma(i),t}+w_s^i$, we obtain the desired form.

Case 2. There is a waiting time at machine M_s for some $1\le s\le m-1$ and at M_m.

The proof that the cycle reaches an $\bar{E}_m$ state at machine M_m is similar to that for M_s in Case 1. Starting from that state with part $P_{\sigma(i)}$ at M_m, cycle π_j includes the following activities: move to I ($\sum_{q=1}^{m}\delta_q-(m-1)\eta$), pick up a new part $P_{\sigma(i+1)}$ (ϵ_1), move to M_1 (δ_1), load $P_{\sigma(i+1)}$ onto M_1 (ϵ_2), wait at M_1 ($p_{\sigma(i+1),1}$), ..., load $P_{\sigma(i+1)}$ on M_s (ϵ_{2s}), move from M_s to M_m ($\sum_{q=s+1}^{m}\delta_q-(m-s-1)\eta$), wait (if necessary) at M_m (w_m^i), unload (ϵ_{2m+1}), move to O (δ_{m+1}), drop $P_{\sigma(i)}$ at O (ϵ_{2m+2}), move to M_s ($\sum_{q=s+1}^{m+1}\delta_q-(m-s)\eta$), wait (if necessary) at M_s (w_s^{i+1}), unload

(ϵ_{2s+1}), move to M_{s+1} (δ_{s+1}), load $P_{\sigma(i+1)}$ onto M_{s+1} (ϵ_{2s+2}), wait at M_{s+1} ($p_{\sigma(i+1),s+1}$), ..., and load $P_{\sigma(i+1)}$ on M_m (ϵ_{2m}). Thus, we have

$T^m_{j\sigma(i)\sigma(i+1)} = \sum_{q=1}^{2m+2} \epsilon_q + 2\sum_{q=1}^{m+1} \delta_q + 2\sum_{q=s+1}^{m} \delta_q + (-3m + 2s + 2)\eta + w^i_m + w^{i+1}_s + \sum_{t=1}^{s-1} p_{\sigma(i+1),t} + \sum_{t=s+1}^{m-1} p_{\sigma(i+1),t}$, where

$w^i_m = \max\{0, p_{\sigma(i),m} - \sum_{q=1}^{2s} \epsilon_q - 2\sum_{q=1}^{m} \delta_q - \sum_{t=1}^{s-1} p_{\sigma(i+1),t} + (2m - s - 2)\eta\}$,

$w^{i+1}_s = \max\{0, p_{\sigma(i+1),s} - 2\sum_{q=s+1}^{m+1} \delta_q - \epsilon_{2m+1} - \epsilon_{2m+2} - w^i_m + (2m - 2s - 1)\eta\}$.

Substituting for w^{i+1}_s, and then for w^i_m, we obtain

$$\begin{aligned} T^m_{j\sigma(i)\sigma(i+1)} \;=\; & \sum_{q=1}^{2m} \epsilon_q + 2\sum_{q=1}^{m} \delta_q - m\eta + \eta + \sum_{t=s+1}^{m-1} p_{\sigma(i+1),t} + \\ & \max\{2\sum_{q=s+1}^{m+1} \delta_q + \epsilon_{2m+1} + \epsilon_{2m+2} - (2m - 2s - 1)\eta + \\ & \sum_{t=1}^{s-1} p_{\sigma(i+1),t}, p_{\sigma(i+1),s} + \sum_{t=1}^{s-1} p_{\sigma(i+1),t}, p_{\sigma(i),m} - (\sum_{q=1}^{2s} \epsilon_q + \\ & 2\sum_{q=1}^{m} \delta_q - (2m - s - 2)\eta) + (2\sum_{q=s+1}^{m+1} \delta_q + \epsilon_{2m+1} + \\ & \epsilon_{2m+2} - (2m - 2s - 1)\eta)\}. \end{aligned}$$

Letting $e_{\sigma(i+1)} = \max\{2\sum_{q=s+1}^{m+1} \delta_q + \epsilon_{2m+1} + \epsilon_{2m+2} - (2m - 2s - 1)\eta + \sum_{t=1}^{s-1} p_{\sigma(i+1),t}, p_{\sigma(i+1),s} + \sum_{t=1}^{s-1} p_{\sigma(i+1),t}\}$ and $f_{\sigma(i)} = p_{\sigma(i),m} - \sum_{q=1}^{2s} \epsilon_q + \epsilon_{2m+1} + \epsilon_{2m+2} - 2\sum_{q=1}^{m} \delta_q + 2\sum_{q=s+1}^{m+1} \delta_q + (s - 1)\eta$, we obtain the desired form.

Finally, we note that the Gilmore-Gomory algorithm can be implemented in time $O(n \log n)$ (Appendix B). ∎

6.6.3 Class *V*2: NP-Hard TSP Problems

MPS cycles belonging to this class are called $V2$-cycles. The associated CRM sequences in these cycles satisfy Condition C3, but not both Conditions C2a and C2b. We will show that in $V2$-cycles the part

scheduling problems associated with these CRM sequences are strongly NP-hard. However, each of these problems can be formulated as a TSP, which provides significant computational advantages as a result of the numerous optimal and heuristic approaches available for that problem (see e.g., Lawler et al. [103], Gutin and Punnen [73]). For example, the heuristic developed by Gendreau et al. [65] can routinely solve a TSP of up to 300 cities (equivalently here, parts) to within 5% of optimality.

Before presenting the following result, we need a definition. It follows from the theory of computational complexity (see e.g., Garey and Johnson [57]) that if the TSP can be solved in polynomial time, then so can an NP-complete part scheduling problem of the type considered here. However, such a solution may require a chain of elaborate formulations and transformations that might be difficult to implement. Our definition that follows is both more restrictive and more practical. We say that a part scheduling problem *can be formulated as a TSP* if and only if the contribution to the cycle time from processing part $P_{\sigma(i)}$ immediately before part $P_{\sigma(i+1)}$ is independent of the data for all other jobs.

LEMMA 6.21 *For any MPS cycle and the associated CRM sequence which does not satisfy Condition C1 in an m-machine cell, the associated part scheduling problem can be formulated as a TSP if and only if there is some* $q, 1 < q \leq m$, *for which the MPS cycle reaches state* $\bar{E}_q$ *at machine* M_q.

Proof. We consider each part of the lemma in turn.
($\Rightarrow$) Assume that the MPS cycle reaches state $\bar{E}_q$ at machine M_q. Let Z_i (resp., Z_{i+1}) denote the time at which the robot loads part $P_{\sigma(i)}$ (resp., part $P_{\sigma(i+1)}$) on M_q. Thus, part $P_{\sigma(i-1)}$ must have left machine M_m by time Z_i, and part $P_{\sigma(i+2)}$ must not have reached machine M_1 by time Z_{i+1}. Consequently, $Z_{i+1} - Z_i$ is independent of the processing times of parts other than $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$. It follows that $Z_{i+1} - Z_i$ can be interpreted as the distance from city i to city $(i+1)$ in a formulation of the TSP.
($\Leftarrow$) Consider an MPS cycle that does not satisfy Condition C1. Since state $\bar{E}_q$ is never reached when part $P_{\sigma(i)}$ is loaded on any machine M_q, either part $P_{\sigma(i-1)}$ or part $P_{\sigma(i+1)}$ is on machine M_j for some j, $1 \leq j \leq m$. In the first case, the remaining processing times in the cell at time Z_{i+1} are not, in general, the same as the remaining processing

times in the cell at time Z_i, because they depend on the processing times of parts $P_{\sigma(i-1)}, P_{\sigma(i)}$, and $P_{\sigma(i+1)}$. In the second case, a similar argument applies for parts $P_{\sigma(i)}, P_{\sigma(i+1)}$, and $P_{\sigma(i+2)}$. In either case, we cannot write $Z_{i+1} - Z_i$ only as a function of the processing times of parts $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$, as required by our definition of a TSP formulation. ■

Lemma 6.21 shows that the part scheduling problem associated with any MPS cycle satisfying Condition C3 can be formulated as a TSP. We will compute the number of such cycles in an m-machine cell. To do so, we need the preliminary results given in the two lemmas that follow.

LEMMA 6.22 *In an m-machine cell, there exists an MPS cycle that does not satisfy Condition C1 and that reaches state $\bar{E}_q$ for some q, $1 < q \leq m$ if and only if there are $\mathcal{N}^{q-1}$ waiting times on machines $M_1, \ldots, M_{q-1}$, one on machine M_q, and $\mathcal{N}^{q-1} - 1$ on machines $M_{q+1}, \ldots, M_m$, where $\mathcal{N}^{q-1} \in \{1, \ldots, \lfloor m/2 \rfloor\}$.*

Proof. Consider an arbitrary MPS cycle π that reaches state $\bar{E}_q$ at machine M_q and that has $\mathcal{N} \geq 1$. By definition of $\bar{E}_q$, there is a waiting time at M_q. Also by definition of $\bar{E}_q$, the robotic cell reaches state $\bar{E}_q$ with the robot at machine M_q in a cycle if and only if the following five events occur in the order shown:
E1: The robot loads part $P_{\sigma(i)}$ onto M_q.
E2: The robot loads part $P_{\sigma(i+1)}$ onto M_1.
E3: The robot drops part $P_{\sigma(i)}$ at O.
E4: The robot loads part $P_{\sigma(i+1)}$ onto M_q.
E5: The robot loads part $P_{\sigma(i+2)}$ onto M_1.
We consider each part of the lemma in turn.

($\Leftarrow$) We consider an MPS cycle that does not satisfy Condition C1. If parts $P_{\sigma(i)}$ and $P_{\sigma(i+2)}$ are on two of machines $M_1, \ldots, M_m$, at any time during this cycle, then Event E5 will precede Event E3. Therefore, a necessary condition to reach state $\bar{E}_q$ at M_q is that there is never more than one other part on machines $M_1, \ldots, M_m$, simultaneously with part $P_{\sigma(i+1)}$. We consider two cases.

Case 1. $\mathcal{N}_{q+1} \geq \mathcal{N}^{q-1}$. Consider the cell immediately after Event E1. From Sethi et al. [142], as discussed in Chapter 3, there are m activities – the loadings on machines $M_1, \ldots, M_{q-1}, M_{q+1}, \ldots, M_m$, and an unloading from machine M_m – between Event E1 and Event E4. Since

Event E3 must precede Event E4, part $P_{\sigma(i)}$ must be advanced at least $m - q + 1$ times during those m activities. Since only parts $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ are on machines $M_1, \ldots, M_m$, $P_{\sigma(i)}$ is advanced $\mathcal{N}^{q-1}$ times from the machines, among $M_1, \ldots, M_{q-1}$, that have waiting times, not from machine M_q that has a waiting time, and $m - q - \mathcal{N}_{q+1}$ times from the machines, among $M_{q+1}, \ldots, M_m$, that do not have waiting times. Thus, $\mathcal{N}^{q-1} + (m - q - \mathcal{N}_{q+1}) \geq m - q + 1$, which is a contradiction.

Case 2. $\mathcal{N}_{q+1} \leq \mathcal{N}^{q-1} - 2$. Since it is necessary for Event E4 to precede Event E5, $P_{\sigma(i+1)}$ must be advanced at least $q - 1$ times during the m activities between Event E2 and Event E5. Since only parts $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ are on machines $M_1, \ldots, M_m$, $P_{\sigma(i+1)}$ is advanced $q - \mathcal{N}^{q-1} - 1$ times from the machines among $M_1, \ldots, M_{q-1}$ that do not have waiting times, once from machine M_q that has a waiting time, and $\mathcal{N}_{q+1}$ times from the machines among $M_{q+1}, \ldots, M_m$ that have waiting times. Thus, $(q - \mathcal{N}^{q-1} - 1) + 1 + \mathcal{N}_{q+1} \geq q - 1$, which is a contradiction.

It therefore follows that $\mathcal{N}_{q+1} = \mathcal{N}^{q-1} - 1$, and the total number of waiting times is $1 + \mathcal{N}^{q-1} + \mathcal{N}_{q+1} = 2\mathcal{N}^{q-1}$. Thus, $\mathcal{N}$ is even. Since there is a waiting time at M_q, it follows that $\mathcal{N}^{q-1} \geq 1$. Finally, since the number of waiting times cannot exceed the number of machines, it follows that $\mathcal{N}^{q-1} \leq \lfloor m/2 \rfloor$.

($\Rightarrow$) It follows from the above discussion that if $\mathcal{N}_{q+1} = \mathcal{N}^{q-1} - 1$, then events E1-E5 will occur in the specified order. Therefore, the robotic cell will reach state $\bar{E}_q$. ∎

COROLLARY 6.1 *Condition C2a implies Condition C3.*

Proof. It follows immediately from the ($\Rightarrow$) part of Lemma 6.22 that if an MPS cycle satisfies Condition C2a with waiting times at machines M_s and M_q, where $1 \leq s < q \leq m$, then the cycle reaches state $\bar{E}_q$ at machine M_q. ∎

It is interesting to note that the converse of Corollary 6.1 is not true (see Table 6.2). For example, MPS cycles with $CRM(\pi_{9,4})$ and $CRM(\pi_{10,4})$ have an $\bar{E}_q$ state, and thus satisfy Condition C3, but have $\mathcal{N} = 4$, which does not satisfy Condition C2a.

LEMMA 6.23 *For a given selection of $\mathcal{N}$ waiting times, $\mathcal{N} \geq 2$ and $\mathcal{N}$ even, in an m-machine cell, there is exactly one MPS cycle that reaches state $\bar{E}_q$ at machine M_q, where $1 < q \leq m$.*

Proof. First, note that the condition in the lemma implies that the conditions in Lemma 6.22 are satisfied at machine M_q for some q, $1 < q \leq m$. Therefore, it follows from the ($\Rightarrow$) part of Lemma 6.22 that there exists at least one MPS cycle satisfying the condition in the lemma.

As discussed in Section 6.3, an MPS cycle with a CRM sequence in an m-machine cell is uniquely characterized by its sequence of $(m+1)$ activities. Therefore, it is sufficient to show that a given selection of $\mathcal{N}$ waiting times among the m machines uniquely determines a sequence of these activities. Assume that there are two distinct MPS cycles, π and π', which both reach state $\bar{E}_q$ with the robot having just completed loading part $P_{\sigma(i)}$ onto machine $M_q, 1 < q \leq m$, and which have waiting times on the same subset of machines. We prove that π and π' are identical by induction on the number of robot moves after $\bar{E}_q$.

By definition of $\bar{E}_q$, the robot next picks up part $P_{\sigma(i+1)}$ and loads it onto M_1 in both cycles π and π'. For the basis of the induction proof, if M_1 has a waiting time in π and π', then the second robot activity after $\bar{E}_q$ in both cycles is to load part $P_{\sigma(i)}$ onto machine M_{q+1}, where we let $M_{m+1} = O$. Otherwise, the second robot activity after $\bar{E}_q$ in both cycles is to load part $P_{\sigma(i+1)}$ onto machine M_2. Thus, the states of the cell are identical in π and π' after two activities. For the induction hypothesis, we assume that the states of the cell are identical in π and π' after t, $2 \leq t \leq m$, robot activities following $\bar{E}_q$. During activity $t+1$, when the robot loads a part ($P_{\sigma(i)}$ or $P_{\sigma(i+1)}$), it either waits during its processing (in both π and π') or unloads a part from another machine. It follows, from the discussion in the ($\Leftarrow$) part of Lemma 6.22 that there is at most one other part ($P_{\sigma(i+1)}$ or $P_{\sigma(i)}$) on any machine. Moreover, from the induction hypothesis, that part is on the same machine in π and π'. Thus, activity $t+1$ is identical in π and π' for $t+1 \leq m+1$. When $t+1 = m+1$, the cycle reaches the same $\bar{E}_q$ state in π and π', and the above arguments can be repeated for subsequent parts. It follows that π and π' are identical. ∎

We have shown in Lemma 6.22 that MPS cycles, based on CRM sequences that reach state $\bar{E}_q$ with the robot at M_q, have a specific number of waiting times at the machines before and after M_q in the cell. From Lemma 6.23, any selection of these waiting times among the m machines defines exactly one MPS cycle that satisfies Condition C3. The following result shows how many such cycles there are.

LEMMA 6.24 *In an m-machine cell, the number of MPS cycles which do not satisfy Condition C1 and in which the associated part scheduling problem can be formulated as a TSP is*

$$\sum_{t=1}^{\lfloor \frac{m}{2} \rfloor} \binom{m}{2t}.$$

Proof. It follows from Lemma 6.21 and the proof of Lemma 6.22 that in any MPS cycle formed by a CRM sequence that does not satisfy Condition C1 and that has an associated part scheduling problem that can be formulated as a TSP, $\mathcal{N}$ is even and $\mathcal{N} \geq 2$. Thus, $\mathcal{N} = 2t$ for some $t \in \{1, \ldots, \lfloor m/2 \rfloor\}$. It follows from Lemma 6.23 that for each possible value of t the number of MPS cycles is

$$\binom{m}{2t},$$

from which the result follows. ∎

It is important to note that the $(2m - 3)$ $V1$-cycles discussed in Lemma 6.20 are included in this number.

We now consider the complexity of part scheduling problems associated with $V2$-cycles. Hall et al. [75] show that the part scheduling problem corresponding to the MPS cycle with $CRM(\pi_{1,3})$ is schedule independent. Furthermore, they show that the part scheduling problems corresponding to the MPS cycles with $CRM(\pi_{3,3})$, $CRM(\pi_{4,3})$, and $CRM(\pi_{5,3})$ are solvable in polynomial time. These cycles satisfy Conditions C2a and C2b and therefore belong to Class $V1$. The part scheduling problems corresponding to the MPS cycles with $CRM(\pi_{2,3})$ and $CRM(\pi_{6,3})$ do not satisfy Condition C3, and therefore they belong to Class W. Thus, there are no $V2$-cycles in a three-machine cell. For this reason it is necessary to study four-machine cells to identify a pattern for larger cells.

In a four-machine cell there are seven MPS cycles that belong to Class V. These are MPS cycles with $CRM(\pi_{4,4})$, $CRM(\pi_{9,4})$, $CRM(\pi_{10,4})$, $CRM(\pi_{11,4})$, $CRM(\pi_{13,4})$, $CRM(\pi_{14,4})$, and $CRM(\pi_{17,4})$, as described in Appendix A. Note that since they satisfy Conditions C2a and C2b, the MPS cycles with $CRM(\pi_{4,4})$, $CRM(\pi_{11,4})$, $CRM(\pi_{13,4})$,

$CRM(\pi_{14,4})$, and $CRM(\pi_{17,4})$ are $V1$-cycles. We show in this section that the part scheduling problems in $V2$-cycles with $CRM(\pi_{9,4})$ and $CRM(\pi_{10,4})$ are strongly NP-hard. All the hardness results in this section are proved for the special case $\delta_i = \delta$ for $i = 0, \ldots, m+1$, $\epsilon_i = \epsilon$ for $i = 1, \ldots, 2m+2$, and $\eta = 0$.

The cycle time for the production of n parts in schedule σ under the MPS cycle with $CRM(\pi_{9,4})$ is derived as follows: $T^3_{9\sigma(i)\sigma(i+1)} = 12\delta + 10\epsilon + a_{\sigma(i+1)} + d_{\sigma(i)} + w_2^{i+1} + w_3^i$, where $w_2^{i+1} = \max\{0, b_{\sigma(i+1)} - d_{\sigma(i)} - w_3^i - 6\delta - 4\epsilon\}$ and $w_3^i = \max\{0, c_{\sigma(i)} - a_{\sigma(i+1)} - 6\delta - 4\epsilon\}$. Since $w_2^{i+1} = \max\{0, b_{\sigma(i+1)} - d_{\sigma(i)} - w_3^i - 6\delta - 4\epsilon\}$, we have

$$w_2^{i+1} + w_3^i = \max\{0, c_{\sigma(i)} - a_{\sigma(i+1)} - 6\delta - 4\epsilon, b_{\sigma(i+1)} - d_{\sigma(i)} - 6\delta - 4\epsilon\}.$$

Therefore, $T_{9(\sigma)} = \sum_{i=1}^n T^3_{9\sigma(i)\sigma(i+1)}$, where $T^3_{9\sigma(i)\sigma(i+1)} = \max\{a'_{\sigma(i+1)} + d'_{\sigma(i)}, c'_{\sigma(i)} + d'_{\sigma(i)}, a'_{\sigma(i+1)} + b'_{\sigma(i+1)}\}$, $a'_i = a_i + 6\delta + 5\epsilon$, $b'_i = b_i + \epsilon$, $c'_i = c_i + \epsilon$ and $d'_i = d_i + 6\delta + 5\epsilon$. This gives

$$T^3_{9\sigma(i)\sigma(i+1)} = a'_{\sigma(i+1)} + d'_{\sigma(i)} + \max\{0, c'_{\sigma(i)} - a'_{\sigma(i+1)}, b'_{\sigma(i+1)} - d'_{\sigma(i)}\}.$$

Note that the distance between cities (parts) i and $(i+1)$ depends only on the parameters of those cities. Thus, we have the required TSP formulation.

LEMMA 6.25 *The part scheduling problem corresponding to an MPS cycle with $CRM(\pi_{9,4})$ is strongly NP-hard.*

Proof. The proof is by reduction from the classical scheduling problem $F_3|$*no-wait*$|C_t$. Röck [136] shows that the cycle time minimization problem is strongly NP-hard. For more details, the reader may refer to the paper on no-wait scheduling by Hall and Sriskandarajah [77]. Let e_i, f_i, and g_i denote the processing time of job i on the first, second, and third machine, respectively, in the three-machine no-wait flowshop. In the three-machine no-wait flowshop, the cycle time $C_{t(\sigma)}$ for the production of n jobs in schedule σ can be written as follows: $C_{t(\sigma)} = \sum_{i=1}^n h_{\sigma(i)\sigma(i+1)}$, where $h_{\sigma(i)\sigma(i+1)} = \max\{e_{\sigma(i)} + f_{\sigma(i)} - e_{\sigma(i+1)}, e_{\sigma(i)} + f_{\sigma(i)} + g_{\sigma(i)} - e_{\sigma(i+1)} - f_{\sigma(i+1)}, e_{\sigma(i)}\}$ (see Appendix B.2 for details). Simplifying, we have $h_{\sigma(i)\sigma(i+1)} = e_{\sigma(i)} + f_{\sigma(i)} - e_{\sigma(i+1)} + \max\{0, g_{\sigma(i)} - f_{\sigma(i+1)}, e_{\sigma(i+1)} - f_{\sigma(i)}\}$. Now, if we set $a'_i = f_i$, $b'_i = e_i$, $c'_i = g_i$, and $d'_i = f_i$, it is clear that minimizing $T_{9(\sigma)}$ is equivalent to minimizing $C_{t(\sigma)}$. ∎

The cycle time for the production of n parts in schedule σ under the MPS cycle with $CRM(\pi_{10,4})$ is derived as follows: $T^3_{10\sigma(i)\sigma(i+1)} = \lambda + \gamma + w_1^{i+1} + w_2^{i+1} + w_3^i + w_4^i$, where $w_1^{i+1} = \max\{0, a_{\sigma(i+1)} - w_3^i - \lambda\}$, $w_2^{i+1} = \max\{0, b_{\sigma(i+1)} - w_4^i - \lambda\}$, $w_3^i = \max\{0, c_{\sigma(i)} - \lambda\}$, $w_4^i = \max\{0, d_{\sigma(i)} - w_1^{i+1} - \lambda\}$, $\lambda = 6\delta + 2\epsilon$, and $\gamma = 12\delta + 8\epsilon$. Therefore,

$$T_{10(\sigma)} = \sum_{i=1}^{n} T^3_{10\sigma(i)\sigma(i+1)} = n(\lambda + \gamma) + \sum_{i=1}^{n}(w_1^{i+1} + w_2^{i+1} + w_3^i + w_4^i).$$

It is easy to see that

$$w_2^{i+1} + w_4^i + w_1^{i+1} + w_3^i = \max\{0, c_{\sigma(i)} - \lambda, b_{\sigma(i+1)} - \lambda, b_{\sigma(i+1)} + c_{\sigma(i)} - 2\lambda, a_{\sigma(i+1)} - \lambda, a_{\sigma(i+1)} + b_{\sigma(i+1)} - 2\lambda, d_{\sigma(i)} - \lambda, c_{\sigma(i)} + d_{\sigma(i)} - 2\lambda\}.$$

LEMMA 6.26 *The part scheduling problem in an MPS cycle with CRM $(\pi_{10,4})$ is strongly NP-hard.*

Proof. The proof is by reduction from problem $F_3|no\text{-}wait|C_t$, and is similar to that of Lemma 6.25. By setting $a_i = e_i + \lambda$, $b_i = f_i + \lambda$, $c_i = f_i + \lambda$ and $d_i = g_i + \lambda$, we have $T^3_{10\sigma(i)\sigma(i+1)} = \lambda + \gamma + f_{\sigma(i)} + f_{\sigma(i+1)} + \max\{0, e_{\sigma(i+1)} - f_{\sigma(i)}, g_{\sigma(i)} - f_{\sigma(i+1)}\}$. Thus, minimizing $T_{10(\sigma)}$ is equivalent to minimizing $C_{t(\sigma)}$. ∎

The following two results relate the complexity of part scheduling problems in cells with m machines to that in larger cells. We consider two cases. The first case requires that $\mathcal{N} < m+1$ in the larger cell, and is studied in Lemma 6.27. The second case requires that $\mathcal{N} = m+2$ in the larger cell, and is studied in Lemma 6.28.

LEMMA 6.27 *If the part scheduling problems associated with all $V2$-cycles in an m-machine cell, $m \geq 4$, are strongly NP-hard, then the part scheduling problems associated with all $V2$-cycles which have $\mathcal{N} < m+1$ in an $(m+1)$-machine cell are strongly NP-hard.*

Proof. Consider an arbitrary $V2$-cycle with $CRM(\pi_{g,m+1})$, where the cycle $\pi_{g,m+1}$ has $\mathcal{N} < m+1$ waiting times in an $(m+1)$-machine cell, $m \geq 4$. Let $M_{j_1}, \ldots, M_{j_{\mathcal{N}}}$ denote the machines at which $\pi_{g,m+1}$ has waiting times. Note that if $\mathcal{N} = 2$, then $2 \leq j_1, j_2 \leq m$, since otherwise the MPS cycle with $CRM(\pi_{g,m+1})$ would be a $V1$-cycle. As in Appendix A, let $\pi_{g,m+1} = \{M^+_{m+1}, \ldots, M^-_p, M^-_q, M^-_r, \ldots, M^+_{m+1}\}$.

Since $\mathcal{N} < m+1$, there exists a machine M_q in $\pi_{g,m+1}$, where $q \in \{1,\ldots,m+1\}\backslash\{j_1,\ldots,j_{\mathcal{N}}\}$, without a waiting time. If $\mathcal{N}=2$, then since $m+1 \geq 5$, there exists a machine $M_q, q \neq 1, q \neq m+1$, satisfying the above condition. Since the robot stays with the part between loading it on M_q and unloading it, we have $r = q+1$. Thus, M_{r-2}^- occurs between M_{m+1}^+ and M_p^- in $\pi_{g,m+1}$. We remove M_q^- from $\pi_{g,m+1}$ to obtain $\pi_{g,m} = \{M_{m+1}^+,\ldots,M_p^-,M_r^-,\ldots,M_{m+1}^+\}$. We use $\pi_{g,m}$ in constructing an appropriate sequence for an m-machine cell.

Consider an m-machine cell containing machines $M_1,\ldots,M_{q-1},M_{q+1},\ldots,M_{m+1}$, in which $M_{q+1},\ldots,M_{m+1}$ are reindexed as $M_q,\ldots,M_m$, respectively. As a result of this reindexing, if $\mathcal{N}=2$, then $2 \leq j_1, j_2 \leq m-1$. Let $\pi_{h,m}$ denote a robot move sequence in that cell in which the machines are loaded in the same sequence as in $\pi_{g,m}$. From Sethi et al. [142], that cycle is unique. First, we show that the MPS cycle with $CRM(\pi_{h,m})$ is a feasible $V2$-cycle. Let M_s denote a machine at which an $\bar{E}_s$ state occurs in $\pi_{g,m+1}$. Since there is no waiting time at M_q, it follows that $s \neq q$. Since M_r is empty and M_{r-2} is occupied by a part (where we let $M_0 = I$) when activity M_p^- is performed in $\pi_{g,m+1}$, activity M_r^- can be performed immediately after M_p^- in $\pi_{g,m}$. Thus, the MPS cycle with $CRM(\pi_{g,m})$ is a feasible cycle in which an E_s state occurs at M_s. Since $\pi_{h,m}$ has the same sequence of machine loadings as $\pi_{g,m}$, it is also feasible and it reaches the state E_s at the machine corresponding to machine M_s. Note that $\pi_{h,m}$ and $\pi_{g,m+1}$ have the same number of waiting times. Since (i) $\pi_{h,m}$ has an E_s state and (ii) $2 \leq j_1, j_2 \leq m-1$ if $\mathcal{N}=2$, the MPS cycle with $CRM(\pi_{h,m})$ is a $V2$-cycle.

Using the condition in the statement of the lemma, we can prove the strong NP-hardness of the part scheduling problem in the MPS cycle with $CRM(\pi_{g,m+1})$ by reduction from that associated with the MPS cycle with $CRM(\pi_{h,m})$. In any cycle in an $(m+1)$-machine cell, the cycle time C can be written as $C = Z + W$, where Z is a constant and W is the total robot waiting time. Since Z is a constant, minimizing C is equivalent to minimizing W. Given an arbitrary instance of the part scheduling problem in the MPS cycle with $CRM(\pi_{h,m})$, using data $p'_{i,j}, i=1,\ldots,n, j=1,\ldots,m$, δ', ϵ', and the total robot waiting time W', we construct an instance of the part scheduling problem in the MPS

cycle with $CRM(\pi_{g,m+1})$ as follows:

$$\begin{aligned}
p_{i,j} &= p'_{i,j} + \Delta_j, \quad i = 1, \ldots, n, \quad j = 1, \ldots, q-1; \\
p_{i,q} &= 0, \quad i = 1, \ldots, n; \\
p_{i,j} &= p'_{i,j-1} + \Delta_j, \quad i = 1, \ldots, n, \quad j = q+1, \ldots, m+1; \\
\delta &= \delta'; \\
\epsilon &= \epsilon'; \\
W &= W',
\end{aligned}$$

where Δ_j denotes the difference between the sum of the total travel time and the loading and unloading times from the loading of a part on M_j until the robot returns to M_j in the $(m+1)$-machine cell, and the corresponding total time in the m-machine cell. Note that Δ_j can be found in time $O(m)$ for $j = 1, \ldots, m$; therefore the above construction requires polynomial time.

We show that the robot waiting time at a machine (except M_q) for a part $P_{\sigma(i)}$ in $\pi_{g,m+1}$ is equal to the waiting time at the corresponding machine for that part in $\pi_{h,m}$. Let the waiting time for part $P_{\sigma(i)}$ at M_j in $\pi_{g,m+1}$ be $w^i_j = \max\{0, p_{\sigma(i),j} - t^i_j - W^i_j\}$, where t^i_j is the total robot activity (travel, loading and unloading) time from the loading of part $P_{\sigma(i)}$ on M_j until the robot returns to M_j, and W^i_j is the total robot waiting time on all the machines during this interval. The waiting time $w'^i_{j'}$ on the corresponding machine $M_{j'}$ in $\pi_{h,m}$ is given by $w'^i_{j'} = \max\{0, p'_{\sigma(i),j'} - t'^i_{j'} - W'^i_{j'}\}$, where the definitions of $t'^i_{j'}$ and $W'^i_{j'}$ are analogous to those of t^i_j and W^i_j, respectively. The definition of q and that $\pi_{h,m}$ and $\pi_{g,m+1}$ have the same sequence of machine loadings imply that the waiting time for each part is zero at M_q in $\pi_{g,m+1}$. Thus, the number of robot waiting times in $W'^i_{j'}$ is the same as in W^i_j. Now, since $t^i_j = t'^i_{j'} + \Delta_j$ from the definition of Δ_j and $p_{\sigma(i),j} = p'_{\sigma(i),j'} + \Delta_j$ by construction, we have $W'^i_{j'} = W^i_j$ and $w^i_j = w'^i_{j'}$. Thus, the part scheduling problems in MPS cycles with $CRM(\pi_{g,m+1})$ and $CRM(\pi_{h,m})$ are equivalent. ∎

LEMMA 6.28 *If the part scheduling problems associated with all $V2$-cycles which have $\mathcal{N} = m$ in an m-machine cell, $m \geq 4$, are strongly NP-hard, then the part scheduling problems associated with all $V2$-cycles*

which have $\mathcal{N} = m+2$ in an $(m+2)$-machine cell are strongly NP-hard.

Proof. From the proof of Lemma 6.22, $\mathcal{N}$ is even in all $V2$-cycles. Therefore, we may assume that m is even. From Lemma 6.23, there is a unique $V2$-cycle in which $\mathcal{N} = m$ in an m-machine cell, where m is even. Using data $p'_{i,j}, i = 1,\ldots,n, j = 1,\ldots,m$, δ', ϵ', and C', its cycle time $T_{h(\sigma)}$ can be developed as follows: $\pi_{h,m} = \{M^-_{m/2+1}, M^-_1, M^-_{m/2+2}, M^-_2, \ldots, M^-_{m/2+j}, M^-_j, \ldots, M^-_m, M^-_{m/2}, M^+_m, M^-_{m/2+1}\}$. This gives $T^{m/2+1}_{h\sigma(i)\sigma(i+1)} = \frac{(m+2)^2}{2}\delta' + (2m+2)\epsilon' + w^{i+1}_1 + \cdots + w^{i+1}_{m/2} + w^i_{m/2+1} + \cdots + w^i_m$, where $w^{i+1}_j = \max\{0, p'_{\sigma(i+1),j} - w^i_{m/2+j} - (m+2)\delta' - 2\epsilon'\}, j = 1,\ldots,m/2; w^i_{m/2+1} = \max\{0, p'_{\sigma(i),m/2+1} - (m+2)\delta' - 2\epsilon'\}$. For $j = 1,\ldots,m/2-1$, $w^i_{m/2+j+1} = \max\{0, p'_{\sigma(i),m/2+j+1} - w^{i+1}_j - (m+2)\delta' - 2\epsilon'\}$. Therefore,

$$\begin{aligned} T^{m/2+1}_{h(\sigma)} &= \sum_{i=1}^{n} T^{m/2+1}_{h\sigma(i)\sigma(i+1)} \\ &= n[\frac{(m+2)^2}{2}\delta' + (2m+2)\epsilon'] \\ &\quad + \sum_{i=1}^{n}(w^{i+1}_1 + \cdots + w^{i+1}_{m/2} + w^i_{m/2+1} + \cdots + w^i_m). \end{aligned}$$

Given an arbitrary instance of the above part scheduling problem in an m-machine cell, we construct an instance of the equivalent problem in an $(m+2)$-machine cell as follows:

$$\begin{aligned} p_{i,1} &= 0, \quad i = 1,\ldots,n; \\ p_{i,j+1} &= p'_{i,j}, \quad i = 1,\ldots,n, \quad j = 1,\ldots,m/2; \\ p_{i,m/2+2} &= 0, \quad i = 1,\ldots,n; \\ p_{i,m/2+j+2} &= p'_{i,j}, \quad i = 1,\ldots,n, \quad j = 1,\ldots,m/2; \\ \delta &= (m+2)\delta'/(m+4); \\ \epsilon &= \epsilon'; \\ C &= C' + (m+2)\delta' + 4\epsilon'. \end{aligned}$$

The cycle time expression for the unique $V2$-cycle with $\mathcal{N} = m+2$ in an $(m+2)$-machine cell, where $(m+2)$ is even, can be developed as follows:

$T^{m/2+2}_{h\sigma(i)\sigma(i+1)} = \frac{(m+4)^2}{2}\delta + (2m+6)\epsilon + w^{i+1}_1 + \cdots + w^{i+1}_{m/2+1} + w^i_{m/2+2} + \cdots + w^i_{m+2}$, where $w^{i+1}_j = \max\{0, p_{\sigma(i+1),j} - w^i_{m/2+j+1} - (m+4)\delta - 2\epsilon\}, \quad j = 1, \ldots, m/2+1$; $w^i_{m/2+2} = \max\{0, p_{\sigma(i),m/2+2} - (m+4)\delta - 2\epsilon\}$; $w^i_{m/2+j+2} = \max\{0, p_{\sigma(i),m/2+j+2} - w^{i+1}_j - (m+4)\delta - 2\epsilon\}, \quad j = 1, \ldots, m/2$.

It is easy to verify that the part scheduling problem associated with the unique $V2$-cycle in which $\mathcal{N} = m$ in the m-machine cell is equivalent to that associated with the unique $V2$-cycle in which $\mathcal{N} = m+2$ in the $(m+2)$-machine cell. ∎

6.6.4 Class *W*: NP-Hard Non-TSP Problems

We show that if a CRM sequence satisfies neither Condition C1 nor Condition C3, in which case we call the corresponding MPS cycle a W-cycle, then the part scheduling problem associated with that cycle is strongly NP-hard. In this kind of cycle, the robot frequently leaves a part to be processed at a machine to perform other activities in the cell. This creates many waiting times and, more importantly, those waiting times are recursive.

It is interesting to note that there is no analog of Lemma 6.23 for W-cycles. That is, the same selection of $\mathcal{N}$ waiting times can appear in more than one W-cycle. For example, the W-cycles with $CRM(\pi_{2,3})$ and $CRM(\pi_{6,3})$, as described in Section 6.3.1, both have waiting times on all the machines in a three-machine cell.

Hall et al. [76] show that the part scheduling problems in MPS cycles with $CRM(\pi_{2,3})$ and $CRM(\pi_{6,3})$ are strongly NP-hard (Section 6.5). The following two results relate the complexity of part scheduling problems in cells with m machines to that in larger cells. We consider two cases. The first case, which requires that $\mathcal{N} < m+1$ in the larger cell, is studied in Lemma 6.29. The second case, which requires that $\mathcal{N} = m+1$ in the larger cell, is studied in Lemma 6.30.

LEMMA 6.29 *If the part scheduling problems associated with all W-cycles in an m-machine cell, $m \geq 3$, are strongly NP-hard, then the part scheduling problems associated with all W-cycles which have $\mathcal{N} < m+1$ in an $(m+1)$-machine cell are strongly NP-hard.*

Proof. First, consider the case $\mathcal{N} = 1$. Here the robot leaves a part, after loading it, only at one machine M_q. Since the robot remains with

the part at all other machines, it can never return and pick up the part at M_q. Consequently, there are no CRM sequences in which $\mathcal{N} = 1$. The contrapositive of Corollary 6.1 implies that there is no W-cycle in which $\mathcal{N} = 2$. We therefore consider an arbitrary W-cycle with $CRM(\pi_{g,m+1})$, where $\pi_{g,m+1}$ has $\mathcal{N}$ waiting times, $3 \leq \mathcal{N} < m + 1$, in an $(m + 1)$-machine cell with $m \geq 3$. Let $M_{j_1}, \ldots, M_{j_\mathcal{N}}$ denote the machines at which $\pi_{g,m+1}$ has waiting times. Let $\pi_{g,m+1} = \{M^+_{m+1}, \ldots, M^-_p, M^-_q, M^-_r, \ldots, M^+_{m+1}\}$. Since $\mathcal{N} < m + 1$, there exists a machine M_q in $\pi_{g,m+1}$ without a waiting time, where $q \in \{1, \ldots, m + 1\} \setminus \{j_1, \ldots, j_\mathcal{N}\}$. Since the robot stays with the part between loading it on M_q and unloading it, we must have $r = q + 1$. Thus, M^-_{r-2} will occur between M^+_{m+1} and M^-_p in $\pi_{g,m+1}$. We remove M^-_q from $\pi_{g,m+1}$ to obtain sequence $\pi_{g,m} = \{M^+_{m+1}, \ldots, M^-_p, M^-_r, \ldots, M^+_{m+1}\}$.

Now consider an m-machine cell containing machines $M_1, \ldots, M_{q-1}$, $M_{q+1}, \ldots, M_{m+1}$, in which $M_{q+1}, \ldots, M_{m+1}$ are re-indexed as $M_q, \ldots, M_m$, respectively. Let $\pi_{h,m}$ denote the unique sequence in that cell in which the machines are loaded in the same sequence as in $\pi_{g,m}$. First, we show that the MPS cycle with $CRM(\pi_{h,m})$ is a feasible W-cycle. Since the MPS cycle with $CRM(\pi_{g,m+1})$ is a W-cycle, it does not contain an $\bar{E}_q$ state for any M_q. Since M_r is empty and M_{r-2} is occupied by a part (where we let $M_0 = I$) when activity M^-_p is performed in $\pi_{g,m+1}$, M^-_r can be performed immediately after M^-_p in $\pi_{g,m}$. Thus, the MPS cycle with $CRM(\pi_{g,m})$ is a feasible cycle which does not contain the state $\bar{E}_q$ for any M_q, since the sequence of machine loadings on the available machines is the same as in $\pi_{g,m+1}$. Therefore, at each machine loading in $\pi_{g,m}$, there is at least one other machine which is occupied by a part. Since $\pi_{h,m}$ has the same sequence of machine loadings as $\pi_{g,m}$, it is also feasible and does not contain an $\bar{E}_q$ state for any M_q. Note that $\pi_{h,m}$ and $\pi_{g,m+1}$ have the same number of waiting times. Since $\pi_{h,m}$ does not contain an $\bar{E}_q$ state, an MPS cycle with $CRM(\pi_{h,m})$ is a W-cycle.

The remainder of the proof is similar to that of Lemma 6.27. ∎

LEMMA 6.30 *If the part scheduling problems associated with all W-cycles which have $\mathcal{N} = m$ in an m-machine cell, $m \geq 3$, are strongly NP-hard, then the part scheduling problems associated with all W-cycles which have $\mathcal{N} = m+1$ in an $(m+1)$-machine cell are strongly NP-hard.*

Proof. Consider an arbitrary W-cycle with $CRM(\pi_{g,m+1})$ in an $(m+1)$-machine cell. The cycle belongs to one of of the following two types:

Type A: $\pi_{g,m+1}$ contains the sequence of machine loadings M_i^-, M_1^-, M_{i+1}^- for some $2 \leq i \leq m+1$, where we let $M_{m+2}^- = M_{m+1}^+$.
Type B: Otherwise.

We first consider an arbitrary Type B cycle $\pi_{g,m+1} = \{M_{m+1}^+, \ldots, M_p^-, M_1^-, M_r^-, \ldots, M_{m+1}^+\}$, where $r \neq p+1$. We let $q = 1$ and remove M_1^- from sequence $\pi_{g,m+1}$ to obtain sequence $\pi_{g,m} = \{M_{m+1}^+, \ldots, M_p^-, M_r^-, \ldots, M_{m+1}^+\}$. Now consider an m-machine cell containing machines $M_2, \ldots, M_{m+1}$, in which those machines are re-indexed as $M_1, \ldots, M_m$, respectively. Let $\pi_{h,m}$ denote the unique sequence in that cell in which the machines are loaded in the same sequence as in $\pi_{g,m}$ and $\mathcal{N} = m$. Thus, $\pi_{h,m} = \{M_m^+, \ldots, M_{p-1}^-, M_{r-1}^-, \ldots, M_m^+\}$. Since removing M_1^- from $\pi_{g,m+1}$ will not create a sequence of the form $\{\ldots, M_i^-, M_{i+1}^-, \ldots\}$ (where we let $M_{m+1}^- = M_m^+$), for some $1 \leq i \leq m$, it follows that $\mathcal{N} = m$ in $\pi_{h,m}$. Since machine M_{r-1} is occupied by a part and machine M_r is free when machine M_p is loaded in $\pi_{g,m+1}$, activity M_r^- can be performed immediately after M_p^- in $\pi_{g,m}$. Thus, $\pi_{g,m}$ is a feasible sequence. Since $\pi_{h,m}$ has the same sequence of machine loadings as $\pi_{g,m}$, it follows that $\pi_{h,m}$ is also a feasible sequence. As in the proof of Lemma 6.29, $\pi_{h,m}$ does not contain an $\bar{E}_q$ state for any M_q and is thus a W-cycle with $\mathcal{N} = m$.

Consider an arbitrary cycle of Type A in an m' machine cell, where $m' \geq 4$. We consider two subtypes, which we denote by A1 and A2.
Type A1: $m' = 4$.

Here we let $m+1 = 4$ and provide a transformation to a cell with $m = 3$ machines. From Appendix A, when $m+1 = 4$, there are only two W-cycles with the following CRM sequences:
$\pi_{20,4} = \{M_4^+, M_4^-, M_2^-, M_1^-, M_3^-, M_4^+\}$,
$\pi_{22,4} = \{M_4^+, M_3^-, M_2^-, M_4^-, M_1^-, M_4^+\}$.

Let $\pi_{g,m+1} : \{M_{m+1}^+, \ldots, M_p^-, M_i^-, M_1^-, M_{i+1}^-, M_r^-, \ldots, M_{m+1}^+\}$ be a CRM sequence of the W-cycle in this class. The following transformation obtains a feasible MPS cycle with $\mathcal{N} = m$ in an m-machine cell. Let $q = i$. Thus, $q = 2$ in $\pi_{20,4}$ and $q = 4$ in $\pi_{22,4}$. We remove M_q^- from sequence $\pi_{g,m+1}$ and interchange the positions of M_{q+1}^- and M_1^- (where we let $M_{m+2}^- = M_{m+1}^+$ in $\pi_{22,4}$). Then the following sequence is obtained:

$\pi_{g,m} : \{M_{m+1}^+, \ldots, M_p^-, M_{i+1}^-, M_1^-, M_r^-, \ldots, M_{m+1}^+\}$. Now consider an m-machine cell containing machines $M_1, \ldots, M_{q-1}, M_{q+1}, \ldots, M_{m+1}$ in which $M_{q+1}, \ldots, M_{m+1}$ are re-indexed as $M_q, \ldots, M_m$, respectively. Let $\pi_{h,m}$ denote the unique sequence in that cell in which the machines are loaded in the same sequence as in $\pi_{g,m}$. When activity M_p^- is performed in $\pi_{g,m+1}$, the following conditions must be satisfied. First, machines M_i and M_{i+1} must be free. Second, machine M_{i-1} must be occupied by a part. Also, by definition of $\pi_{g,m+1}$, $p \neq i-1$. Finally, note that in both $\pi_{20,4}$ and $\pi_{22,4}$ we have $r \neq 2$. Under these conditions, the machine loading sequence $\{M_p^-, M_{i+1}^-, M_1^-, M_r^-\}$ is feasible in $\pi_{g,m}$, and thus $\pi_{g,m}$ is a feasible sequence. Furthermore, since it has the same sequence of machine loadings as $\pi_{g,m}$, the sequence $\pi_{h,m}$ is also feasible. As above, $\pi_{h,m}$ does not contain an $\bar{E}_q$ state for any M_q, and is thus a W-cycle with $\mathcal{N} = m$. Using the above transformation and the following construction, MPS cycles with $CRM(\pi_{6,3})$ and $CRM(\pi_{2,3})$ are reducible to MPS cycles with $CRM(\pi_{20,4})$ and $CRM(\pi_{22,4})$, respectively.

Consider an arbitrary instance of the part scheduling problem of Type A1 or Type B associated with an MPS cycle with $CRM(\pi_{h,m})$. Using data $p'_{i,j}, i = 1, \ldots, n, j = 1, \ldots, m$, δ', and ϵ', and the total robot waiting time W', we construct an instance of the part scheduling problem associated with an MPS cycle with $CRM(\pi_{g,m+1})$ as follows:

$$\begin{aligned}
p_{i,j} &= p'_{i,j} + \Delta_j, \quad i = 1, \ldots, n, \quad j = 1, \ldots, q-1; \\
p_{i,q} &= 0, \quad i = 1, \ldots, n; \\
p_{i,j} &= p'_{i,j-1} + \Delta_j, \quad i = 1, \ldots, n, \quad j = q+1, \ldots, m+1; \\
\delta &= \delta'; \\
\epsilon &= \epsilon'; \\
W &= W',
\end{aligned}$$

where Δ_j is defined in the proof of Lemma 6.27.

The remainder of the proof is similar to that of Lemma 6.27.

Type A2: $m' > 4$.

Here we let $m + 2 = m'$ and provide a transformation to a cell with m machines. Let $\pi_{g,m+2} : \{M_{m+2}^+, \ldots, M_p^-, M_i^-, M_1^-, M_{i+1}^-, M_r^-, \ldots, M_{m+2}^+\}$. By definition, $p \neq i-1$ and $r \neq i+2$. Here, we need the following transformation to obtain a feasible cycle with $\mathcal{N} = m$ in

the m-machine cell. Let $q = i$. We remove M_i^- and M_1^- from sequence $\pi_{g,m+2}$ to obtain $\pi_{g,m} = \{M_{m+2}^+, \ldots, M_p^-, M_{i+1}^-, M_r^-, \ldots, M_{m+2}^+\}$. Consider an m-machine cell with machines $M_2, \ldots, M_{q-1}, M_{q+1}, \ldots, M_{m+2}$, in which $M_2, \ldots, M_{q-1}$ are re-indexed as $M_1, \ldots, M_{q-2}$, respectively, and $M_{q+1}, \ldots, M_{m+2}$ are re-indexed as $M_{q-1}, \ldots, M_m$, respectively. Let $\pi_{h,m}$ denote the unique sequence in that cell in which the machines are loaded in the same sequence as in $\pi_{g,m}$. When activity M_p^- is performed in $\pi_{g,m+2}$, the following conditions must be satisfied. First, machines M_i and M_{i+1} must be free. Second, machine M_{i-1} must be occupied by a part. Under these conditions, activity M_{i+1}^- can be performed immediately after M_p^- in $\pi_{g,m}$; thus $\pi_{g,m}$ is a feasible sequence. Furthermore, since it has the same sequence of machine loadings as $\pi_{g,m}$, the sequence $\pi_{h,m}$ is also feasible. As above, $\pi_{h,m}$ does not contain an $\bar{E}_q$ state for any M_q. Thus, an MPS cycle with $CRM(\pi_{h,m})$ is a W-cycle with $\mathcal{N} = m$.

Consider an arbitrary instance of the part scheduling problem of Type A2 associated with the MPS cycle with $CRM(\pi_{h,m})$. Using data $p'_{i,j}, i = 1, \ldots, n, j = 1, \ldots, m$, δ', and ϵ', and the total robot waiting time W', we construct an instance of the part scheduling problem associated with the MPS cycle with $CRM(\pi_{g,m+2})$ as follows:

$$
\begin{aligned}
p_{i,1} &= 0, \quad i = 1, \ldots, n; \\
p_{i,j} &= p'_{i,j} + \Delta_j, \quad i = 1, \ldots, n, \quad j = 2, \ldots, q-1; \\
p_{i,q} &= 0, \quad i = 1, \ldots, n; \\
p_{i,j} &= p'_{i,j-1} + \Delta_j, \quad i = 1, \ldots, n, \quad j = q+1, \ldots, m+2; \\
\delta &= \delta'; \\
\epsilon &= \epsilon'; \\
W &= W',
\end{aligned}
$$

where Δ_j is defined in the proof of Lemma 6.27.

The remainder of the proof is similar to that of Lemma 6.27. ∎

6.6.5 Overview

The main result of our analysis now follows.

THEOREM 6.15 *In an m-machine robotic cell with $m \geq 4$, there are $m!$ CRM sequences. Out of these*

a. one U-cycle defines a trivially solvable part scheduling problem,

b. $(2m-3)$ *$V1$-cycles define a part scheduling problem which is solvable in time* $O(n \log n)$,

c.

$$\sum_{t=1}^{\lfloor \frac{m}{2} \rfloor} \begin{pmatrix} m \\ 2t \end{pmatrix} - 2m + 3$$

$V2$-cycles define part scheduling problems that can be formulated as TSPs and that are strongly NP-hard, and

d.

$$m! - 1 - \sum_{t=1}^{\lfloor \frac{m}{2} \rfloor} \begin{pmatrix} m \\ 2t \end{pmatrix}$$

W-cycles define part scheduling problems that, in general, cannot be formulated as TSPs (in the sense defined in Section 6.6.3) and that are strongly NP-hard.

Proof. Sethi et al. [142] show that there are $m!$ 1-unit robot move sequences in an m-machine cell (Chapter 3, Appendix A).

a. Follows from Lemma 6.19.

b. Follows from Lemma 6.20.

c. The $(\Rightarrow)$ part of Lemma 6.21 shows that the associated part scheduling problems can each be formulated as a TSP. The proof of NP-hardness is by induction on m. Lemmas 6.25 and 6.26 provide the basis for $m = 4$. Lemma 6.27 provides the induction step for $\mathcal{N} < m + 1$. Since by assumption there exists a $V2$-cycle with $\mathcal{N} = m$, and since from the proof of Lemma 6.22 $\mathcal{N}$ is even in all $V2$-cycles, there are no $V2$-cycles with $\mathcal{N} = m + 1$. Lemma 6.28 provides the induction step for $\mathcal{N} = m + 2$. The number of cycles follows from Lemmas 6.20 and 6.24.

d. The lack of a TSP formulation follows from the $(\Leftarrow)$ part of Lemma 6.21. The proof of NP-hardness is by induction on m. Section 6.5 provides the basis for $m = 3$. Lemma 6.29 provides the induction step for $\mathcal{N} < m + 1$. Lemma 6.30 provides the induction step for $\mathcal{N} = m + 1$. The number of cycles follows from the formula for the total number of cycles in Sethi *et al.* [142] and from the earlier parts of the theorem. ∎

Note that Sethi et al. [142] (resp., Hall et al. [75, 76]) resolve the complexity of the part scheduling problems associated with the two (resp., six) CRM sequences in a two- (resp., three-) machine cell (Sections 6.2, 6.3.2, and 6.5). Therefore, those results and Theorem 6.15 resolve the

complexity of the part scheduling problems associated with all CRM sequences in any cell. It is interesting to note, as discussed in Section 6.6.3, that there are no $V2$-cycles in cells with fewer than four machines. This is the reason why the basis of the induction proof uses $m = 4$ in part c and $m = 3$ in part d of Theorem 6.15.

As an illustration of Theorem 6.15, for the $m! = 24$ robot move sequences in a four-machine cell (Appendix A), one U-cycle has an associated part scheduling problem that is schedule independent; seven V-cycles have associated part scheduling problems that can be formulated as TSPs (of these, five are $V1$-cycles that are solvable by the Gilmore-Gomory algorithm and the remaining two are strongly NP-hard $V2$-cycles); and 16 W-cycles have strongly NP-hard associated part scheduling problems without a natural TSP structure (in the sense defined in Section 6.6.3).

Finally, it is interesting to note that the recognition versions of the part scheduling problems associated with the NP-hard $V2$- and W-cycles are in the class NP, since given a part schedule, both the cycle time and the robot waiting times can be calculated using linear programming.

6.7 Heuristics for Three-Machine Problems

For three machine cells, this section designs and tests simple heuristics for those part scheduling problems (corresponding to CRM sequences) that are intractable. This then enables us to develop a heuristic for a completely general three-machine cell (Section 6.7.4). Extensions of these ideas to larger cells are provided in Section 6.8.

6.7.1 A Heuristic Under the Sequence $CRM(\pi_2)$

We develop a heuristic for problem $RF_3|(free,A,MP,CRM(\pi_2))|\mu$. We showed in Section 6.4 that for n even, the MPS cycle with $CRM(\pi_2)$ consisting of a single MPS defines a steady state. However, for n odd, two MPS cycles are needed to define a steady state. Consequently, we let v denote the number of parts in a cycle in a steady state, where $v = 2n$ if n is odd and $v = n$ if n is even. For n odd, a given part schedule σ means a schedule of $2n$ parts in which the first n parts corresponding to an MPS have the same order as the last n parts. Given σ, let $\bar{T}_{2(\sigma)}$ (resp., $T_{2(\sigma)}$) denote the average cycle time for the production of v (resp., n) parts

under the MPS cycle with $CRM(\pi_2)$. Note that $T_{2(\sigma)} = \frac{n}{v}\bar{T}_{2(\sigma)}$. We now propose a heuristic for this problem. The idea behind the procedure is developed from the following cycle time expression, which is derived in detail in Section 6.3.1:

$$\begin{aligned}\bar{T}_{2(\sigma)} &= \sum_{i=1}^{v} T^2_{2\sigma(i+1)\sigma(i+2)} \\ &= v\alpha_2 + \sum_{i=1}^{v} \max\{\beta_2, a_{2\sigma(i+2)}, b_{2\sigma(i+1)} - w_3^i\} + \sum_{i=1}^{v} w_3^i,\end{aligned}$$

where $w_3^i = \max\{0, c_{2\sigma(i)} - \max\{\beta_2, a_{2\sigma(i+1)} - \max\{0, b_{2\sigma(i)} - \beta_2 - w_3^{i-1}\}\}\}$, $b_{2\sigma(v+1)} = b_{2\sigma(1)}$, $a_{2\sigma(v+1)} = a_{2\sigma(1)}$, $a_{2\sigma(v+2)} = a_{2\sigma(2)}$, $\alpha_2 = 2\delta_2 + 2\delta_3 - \eta$, $\beta_2 = \sum_{i=1}^{8} \epsilon_i + 2\delta_1 + 2\delta_2 + 2\delta_3 + 2\delta_4 - 3\eta$, $a_{2\sigma(i)} = a_{\sigma(i)} + \sum_{i=1}^{4} \epsilon_i + \sum_{i=7}^{8} \epsilon_i + 2\delta_1 + 2\delta_4 - 2\eta$, $b_{2\sigma(i)} = b_{\sigma(i)} + \sum_{i=3}^{6} \epsilon_i$, and $c_{2\sigma(i)} = c_{\sigma(i)} + \sum_{i=1}^{2} \epsilon_i + \sum_{i=5}^{8} \epsilon_i + 2\delta_1 + 2\delta_4 - 2\eta$.

For $v = 2n$, since the schedule of the first n parts is the same as the last n parts, the heuristic considers only the schedule of the first n parts. Note that if $w_3^i = 0$ for $i = 1, \ldots, v$, then $\bar{T}_{2(\sigma)} = v\alpha_2 + \sum_{i=1}^{v} \max\{\beta_2, a_{2\sigma(i+2)}, b_{2\sigma(i+1)}\}$. The application of the Gilmore-Gomory algorithm in Step 2 of the following heuristic usually provides a good solution when the values of w_3^i are either zero or very small. Alternatively, if $w_3^i = c_{2\sigma(i)} - \beta_2$ for $i = 1, \ldots, v$, then $\bar{T}_{2(\sigma)} = v\alpha_2 + \sum_{i=1}^{v} \max\{\beta_2 + w_3^i, a_{2\sigma(i+2)} + w_3^i, b_{2\sigma(i+1)}\}$. In this case, the application of the Gilmore-Gomory algorithm in Step 3 (ignoring the $a_{2\sigma(i+2)} + w_3^i$ term in the expression for $\bar{T}_{2(\sigma)}$) usually provides a good solution. Finally, Step 3 compares the two solutions and selects the one with the smaller cycle time.

Heuristic $MPS(\pi_2)$cycle

Step 0: Given $a_i, b_i, c_i, 1 \le i \le n, \delta_1, \ldots, \delta_4, \epsilon_1, \ldots, \epsilon_8$, and η.
$\alpha_2 = 2\delta_2 + 2\delta_3 - \eta$,
$\beta_2 = \sum_{i=1}^{8} \epsilon_i + 2\sum_{i=0}^{4} \delta_i - 3\eta$.

Step 1: For $i = 1, \ldots, n$, find the following:
$a_{2i} = a_i + \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 + \epsilon_7 + \epsilon_8 + 2\delta_1 + 2\delta_4 - 2\eta$,
$b_{2i} = b_i + \epsilon_3 + \epsilon_4 + \epsilon_5 + \epsilon_6$,
$c_{2i} = c_i + \epsilon_1 + \epsilon_2 + \epsilon_5 + \epsilon_6 + \epsilon_7 + \epsilon_8 + 2\delta_1 + 2\delta_4 - 2\eta$.

Step 2: $e_i = a_{2i}, f_i = \max\{\beta_2, b_{2i}\}, i = 1, \ldots, n$.
Apply the Gilmore-Gomory algorithm to the

instance $\{e_i, f_i\}, i = 1, \ldots, n$, obtain the part schedule σ, and calculate $T_{2(\sigma)}$. If $T_{2(\sigma)} < Z_h$, then $Z_h = T_{2(\sigma)}, \sigma^* = \sigma$.

Step 3: $e_i = b_{2i}, f_i = \max\{\beta_2, c_{2i}\}, i = 1, \ldots, n$.
Apply the Gilmore-Gomory algorithm to the instance $\{e_i, f_i\}, i = 1, \ldots, n$, obtain the part schedule σ', and calculate $T_{2(\sigma')}$. If $T_{2(\sigma')} < Z_h$, then $Z_h = T_{2(\sigma)}, \sigma^* = \sigma'$.
Terminate.

We now develop a mixed integer programming formulation for the problem. We use this formulation to test the performance of the heuristic. From Hall et al. [75],

$$\bar{T}_{2(\sigma)} = v(\alpha_2 + \beta_2) + \sum_{q=1}^{3} \sum_{i=1}^{v} w_q^i,$$

where $w_1^{i+2} = \max\{0, a_{2\sigma(i+2)} - w_2^{i+1} - \beta_2\}, w_2^{i+1} = \max\{0, b_{2\sigma(i+1)} - w_3^i - \beta_2\}$, $w_3^i = \max\{0, c_{2\sigma(i)} - w_1^{i+1} - \beta_2\}$, and $v(\alpha_2 + \beta_2)$ is constant for a given problem instance.

Based on these equations, we can write a formulation associated with $RF_3|(free,A,MP,CRM(\pi_2))|\mu$ as the following mixed integer programming problem (MIP2):

$$\min \sum_{q=1}^{3} \sum_{i=1}^{v} w_q^i$$

subject to:

$$\sum_{j=1}^{n} X_{ij} = 1,\ i = 1, \ldots, n,$$
$$\sum_{i=1}^{n} X_{ij} = 1,\ j = 1, \ldots, n,$$
$$X_{ij} = X_{i+n,j+n},\ i = 1, \ldots, n,\ j = 1, \ldots, n,$$
$$w_1^i \geq \sum_{j=1}^{n} a_{2j} X_{ij} - \beta_2 - w_2^{i-1},\ i = 2, \ldots, v+1,$$
$$w_2^i \geq \sum_{j=1}^{n} b_{2j} X_{ij} - \beta_2 - w_3^{i-1},\ i = 2, \ldots, v+1,$$
$$w_3^i \geq \sum_{j=1}^{n} c_{2j} X_{ij} - \beta_2 - w_1^{i+1},\ i = 1, \ldots, v,$$
$$w_q^i \geq 0,\ i = 1, \ldots, v,\ q = 1, 2, 3,$$
$$X_{ij} \in \{0, 1\},\ i = 1, \ldots, v, j = 1, \ldots, v.$$

Here $X_{ij} = 1$ if job j is scheduled in the ith position of the schedule σ and $X_{ij} = 0$ otherwise, $i = 1, \ldots, v, j = 1, \ldots, v$. Also, in the above formulation, we let $w_1^1 = w_1^{v+1}$ and $w_2^1 = w_2^{v+1}$, and when $v = 2n$ we let $a_{2j} = a_{2,j+n}, b_{2j} = b_{2,j+n}$ and $c_{2j} = c_{2,j+n}, j = 1, \ldots, n$. The third set of constraints is not relevant when $v = n$.

6.7.2 A Heuristic Under the Sequence $CRM(\pi_6)$

We develop a heuristic for problem $RF_3|(free,A,MP,CRM(\pi_6))|\mu$. Note that, whether n is even or odd, the MPS cycle with $CRM(\pi_6)$ consisting of a single MPS is sufficient to define a steady state (Section 6.4). The procedure is based on the following cycle time expression derived in Section 6.3.1:

$$\begin{aligned} T_{6(\sigma)} &= \sum_{i=1}^{n} T^1_{6\sigma(i+1)\sigma(i+2)} \\ &= n\alpha_6 + \sum_{i=1}^{n} \max\{\beta_6, c_{6\sigma(i)} - w_1^{i+1}, b_{6\sigma(i+1)}, a_{6\sigma(i+2)}\}, \end{aligned}$$

where $w_1^{i+1} = \max\{0, a_{6\sigma(i+1)} - \max\{\beta_6, b_{6\sigma(i)}, c_{6\sigma(i-1)} - w_1^i\}\}$, $\alpha_6 = 0$, $\beta_6 = 2\delta_1 + 4\delta_2 + 4\delta_3 + 2\delta_4 + \sum_{i=1}^{8} \epsilon_i - 4\eta$, $a_{6\sigma(i)} = a_{\sigma(i)} + 2\delta_1 + 2\delta_2 + \sum_{i=1}^{4} \epsilon_i - \eta$, $b_{6\sigma(i)} = b_{\sigma(i)} + 2\delta_2 + 2\delta_3 + \sum_{i=3}^{6} \epsilon_i - \eta$, and $c_{6\sigma(i)} = c_{\sigma(i)} + 2\delta_3 + 2\delta_4 + \sum_{i=5}^{8} \epsilon_i - \eta$.

Note that $T_{6(\sigma)} = n\alpha_6 + \sum_{i=1}^{n} \max\{\beta_6, c_{6\sigma(i)} - w_1^i, b_{6\sigma(i+1)}, a_{6\sigma(i+2)}\}$. The use of the Gilmore-Gomory algorithm in Step 2 (ignoring the $c_{6\sigma(i)} - w_1^{i+1}$ term in $T_{6(\sigma)}$) usually provides a good solution when the values of w_1^i are large compared to the values of c_i. Step 3 usually provides a good solution when the values of w_1^i are relatively small. Finally, Step 3 compares the two solutions and selects the one with the smaller cycle time.

Heuristic $MPS(\pi_6)$cycle

Step 0: Given $a_i, b_i, c_i, 1 \le i \le n, \delta_1, \ldots, \delta_4, \epsilon_1, \ldots, \epsilon_8$, and η.
$\alpha_6 = 0$,
$\beta_6 = 2\delta_1 + 4\delta_2 + 4\delta_3 + 2\delta_4 + \sum_{i=1}^{8} \epsilon_i - 4\eta$.

Step 1: $a_{6i} = a_i + \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 + 2\delta_1 + 2\delta_2 - \eta, i = 1, \ldots, n$,
$b_{6i} = b_i + \epsilon_3 + \epsilon_4 + \epsilon_5 + \epsilon_6 + 2\delta_2 + 2\delta_3 - \eta, i = 1, \ldots, n$,
$c_{6i} = c_i + \epsilon_5 + \epsilon_6 + \epsilon_7 + \epsilon_8 + 2\delta_3 + 2\delta_4 - \eta, i = 1, \ldots, n$.

Step 2: $e_i = a_{6i}, f_i = \max\{\beta_6, b_{6i}\}, i = 1, \ldots, n$.
Apply the Gilmore-Gomory algorithm to the instance $\{e_i, f_i\}, i = 1, \ldots, n$, obtain the part schedule σ, and calculate $T_{6(\sigma)}$. If $T_{6(\sigma)} < Z_h$, then $Z_h = T_{6(\sigma)}, \sigma^* = \sigma$.

Step 3: $e_i = b_{6i}, f_i = \max\{\beta_6, c_{6i}\}, i = 1, \ldots, n$.
Apply the Gilmore-Gomory algorithm to the

instance $\{e_i, f_i\}, i = 1, \ldots, n$, obtain the part schedule σ', and calculate $T_{6(\sigma')}$. If $T_{6(\sigma')} < Z_h$, then $Z_h = T_{6(\sigma')}, \sigma^* = \sigma'$. Terminate.

We now develop a mixed integer programming formulation for problem $RF_3|(free,A,MP,CRM(\pi_6))|\mu$ in order to test the performance of the heuristic. Given a part schedule σ, the cycle time for the production of n parts under the MPS cycle with $CRM(\pi_6)$ is given by (see Section 6.3.1)

$$T_{6(\sigma)} = n(\alpha_6 + \beta_6) + \sum_{q=1}^{3} \sum_{i=1}^{n} w_q^i,$$

where $w_1^{i+2} = \max\{0, a_{6\sigma(i+2)} - \beta_6 - w_2^{i+1} - w_3^i\}$, $w_2^{i+1} = \max\{0, b_{6\sigma(i+1)} - \beta_6 - w_3^i\}$, and $w_3^i = \max\{0, c_{6\sigma(i)} - \beta_6 - w_1^{i+1}\}$. Note that $n(\alpha_6+\beta_6)$ is constant for a given problem instance. Based on these, we can formulate $RF_3|(free,A,MP,CRM(\pi_6))|\mu$ as the following mixed integer programming problem (MIP6):

$$\min \sum_{q=1}^{3} \sum_{i=1}^{n} w_q^i$$

subject to:

$$\sum_{j=1}^{n} X_{ij} = 1,\ i = 1, \ldots, n,$$
$$\sum_{i=1}^{n} X_{ij} = 1,\ j = 1, \ldots, n,$$
$$w_1^i \geq \sum_{j=1}^{n} a_{6j} X_{ij} - \beta_6 - w_2^{i-1} - w_3^{i-2},\ i = 3, \ldots, n+2,$$
$$w_2^i \geq \sum_{j=1}^{n} b_{6j} X_{ij} - \beta_6 - w_3^{i-1},\ i = 2, \ldots, n+1,$$
$$w_3^i \geq \sum_{j=1}^{n} c_{6j} X_{ij} - \beta_6 - w_1^{i+1},\ i = 1, \ldots, n,$$
$$w_q^i \geq 0,\ q = 1, 2, 3,\ i = 1, \ldots, n,$$
$$X_{ij} \in \{0, 1\},\ i = 1, \ldots, n,\ j = 1, \ldots, n.$$

Here $X_{ij} = 1$ if job j scheduled in the ith position of the schedule σ and $X_{ij} = 0$ otherwise, $i = 1, \ldots, n, j = 1, \ldots, n$. Also, in the above formulation, we let $w_1^1 = w_1^{n+1}$, $w_1^2 = w_1^{n+2}$, and $w_2^1 = w_2^{n+1}$.

6.7.3 Computational Testing

MIP2 and MIP6 were solved optimally using CPLEX. We consider the following data as fixed: $\epsilon_i = 1.0, i = 1, \ldots, 8, \delta_i = 4.0, i = 1, \ldots, 4$, and η=0. Thus, $a_{2\sigma(i)} = a_{\sigma(i)} + 22.0$, $b_{2\sigma(i)} = b_{\sigma(i)} + 4.0$, $c_{2\sigma(i)} = c_{\sigma(i)} + 22.0$, $a_{6\sigma(i)} = a_{\sigma(i)} + 20.0$, $b_{6\sigma(i)} = b_{\sigma(i)} + 20.0$, and $c_{6\sigma(i)} = c_{\sigma(i)} + 20.0$. The

processing times were integers between 0 and 100 generated uniformly and independently. Thus, $pmax = 100$ is the maximum possible processing time on a machine.

Let C^q (resp., C_H^q) denote the cycle time delivered by MIP2 or MIP6 (resp., $MPS(\pi_2)$cycle or $MPS(\pi_6)$cycle) for instance q. Table 6.3 compares the performance of $MPS(\pi_2)$cycle and MIP2 on 150 randomly generated instances of the problem. The fourth column, Opt., shows the number of optimal solutions (out of 50) found by $MPS(\pi_2)$cycle in each row. The next two columns show the mean relative percentage error (MRPE) and worst (i.e., maximum) relative percentage error (WRPE), respectively, of $MPS(\pi_2)$cycle over the 50 problem instances in each row. These are defined as MRPE $= 100 \sum_{q=1}^{50} (C_H^q - C^q)/50C^q$ and WRPE $= 100\max_{1 \le q \le 50}\{(C_H^q - C^q)/C^q\}$.

Tables 6.3 and 6.4 compare the performance of $MPS(\pi_2)$cycle and $MPS(\pi_6)$cycle with the optimal solutions obtained by solving MIP2 and MIP6, respectively. The fifth column shows the mean relative percentage error (MRPE1), compared to the MIP2 solution in Table 6.3, or the MIP6 solution in Table 6.4, averaged over all 50 instances in each row. The sixth column, MRPE2, shows this average calculated only over instances where an optimal solution was not found. The next column, WRPE, shows the worst relative percentage error found among the 50 instances in each row. As the size of the problem instance increases, the number of optimal solutions found by the heuristic decreases and the mean relative percentage error increases at a modest rate. It is interesting to note, however, that the worst relative percentage error shows no evidence of a similar behavior.

n	$pmax$	No.	Opt.	MRPE1	MRPE2	WRPE
5	100	50	18	2.528	3.950	10.884
7	100	50	2	4.792	4.991	13.808
10	100	50	0	5.539	5.539	10.729

Table 6.3. Performance of $MPS(\pi_2)$cycle *vs.* MIP2.

n	$pmax$	No.	Opt.	MRPE1	MRPE2	WRPE
5	100	50	30	1.686	4.215	12.266
7	100	50	9	3.854	4.700	11.796
10	100	50	4	4.405	4.788	11.334

Table 6.4. Performance of $MPS(\pi_6)$cycle *vs.* MIP6.

6.7.4 Heuristics for General Three-Machine Problems

The problem studied here is a general three-machine robotic cell $(RF_3|(free,A,MP,cyclic\text{-}n)|\mu)$. Thus, an optimal solution is identified over all possible robot move sequences. In Section 6.5, we showed that the recognition version of this problem is strongly NP-complete. Consequently, it is unlikely that we can find an optimal solution, even for instances of modest sizes. However, we provide a reasonably effective heuristic. Recall that the six 1-unit robot move sequences, $\pi_1, \ldots, \pi_6$, that are available in this cell define a part scheduling problem that we already know how to solve, either optimally by the results of Hall et al. [75] or heuristically using $MPS(\pi_2)$cycle and $MPS(\pi_6)$cycle. We begin by identifying fast (polynomial-time) heuristics for the part scheduling problem in two MPS cycles, each of which has a robot move sequence which is composed of three 1-unit sequences. To do so, we provide new formulations for these problems. The robot move sequences used are combinations of the 1-unit sequences for which the part scheduling problem is efficiently solvable, as discussed in Section 6.3.2. Let $RM(\pi_1, \pi_3, \pi_4)$ denote the robot move sequence in an MPS that combines 1-unit sequences π_1, π_3, and π_4. Note that the MPS cycle obtained in the following theorems may be given by a non-CRM sequence.

THEOREM 6.16 *Problem $RF_3|(free,A,MP,RM(\pi_1,\pi_3,\pi_4))|\mu$ can be formulated as a TSP (in the sense defined in Section 6.6.3).*

Proof. Note that $E = (\emptyset, \emptyset, \Omega, M_3^-)$ is a common state to each of the sequences π_1, π_3, and π_4. Starting from this common state, where machines M_1 and M_2 are free and the robot has just completed the loading of part $P_{\sigma(i)}$ onto M_3, the robot has three ways in which it can produce a single part and return to the same state. These consist of

following one of the three robot move sequences π_1, π_3, or π_4. The derivation of the following results is similar to that in Section 6.3.1, but uses different starting states.

If π_1 is used, we have

$$T^3_{1\sigma(i)\sigma(i+1)} = \alpha_1 + c_{\sigma(i)} + a_{\sigma(i+1)} + b_{\sigma(i+1)},$$

where $\alpha_1 = 2\sum_{i=1}^{4}\delta_i + \sum_{i=1}^{8}\epsilon_i - 3\eta$.
Alternatively, if π_3 is used, we have

$$T^3_{3\sigma(i)\sigma(i+1)} = \alpha_3 + \max\{\beta_3 + a_{\sigma(i+1)}, b_{3\sigma(i+1)}, a_{\sigma(i+1)}, c_{3\sigma(i)}\},$$

where $\alpha_3 = 2\delta_3+\epsilon_5+\epsilon_6-\eta, \beta_3 = 2\sum_{i=1}^{4}\delta_i+\sum_{i=1}^{4}\epsilon_i+\epsilon_7+\epsilon_8-2\eta, b_{3\sigma(i)} = b_{\sigma(i)} + 2\delta_1 + 2\delta_2 + \sum_{i=1}^{4}\epsilon_i - \eta$, and $c_{3\sigma(i)} = c_{\sigma(i)} + 2\delta_4 + \epsilon_7 + \epsilon_8$.
Finally, if π_4 is used, we have

$$T^3_{4\sigma(i)\sigma(i+1)} = \alpha_4 + b_{\sigma(i+1)} + \max\{\beta_4, c_{4\sigma(i)}, a_{4\sigma(i+1)}\},$$

where $\alpha_4 = 2\delta_2+2\delta_3+\epsilon_3+\epsilon_4+\epsilon_5+\epsilon_6-2\eta, \beta_4 = 2\delta_1+2\delta_2+2\delta_3+2\delta_4+\epsilon_1+\epsilon_2+\epsilon_7+\epsilon_8-3\eta, a_{4\sigma(i)} = a_{\sigma(i)}+2\delta_1+\epsilon_1+\epsilon_2$, and $c_{4\sigma(i)} = c_{\sigma(i)}+2\delta_4+\epsilon_7+\epsilon_8$.

Thus, the cycle time expression becomes

$$T_\sigma = \sum_{i=1}^{n} h_{\sigma(i)\sigma(i+1)},$$

where $h_{\sigma(i)\sigma(i+1)} = \min\{T^3_{1\sigma(i)\sigma(i+1)}, T^3_{3\sigma(i)\sigma(i+1)}, T^3_{4\sigma(i)\sigma(i+1)}\}$ can be interpreted as the distance between city i to city $(i+1)$ in a traveling salesman problem. ∎

We next consider problem $RF_3|(free,A,MP,RM(\pi_1,\pi_4,\pi_5))|\mu$.

THEOREM 6.17 *Problem $RF_3|(free,A,MP,RM(\pi_1,\pi_4,\pi_5))|\mu$ can be formulated as a TSP (in the sense defined in Section 6.6.3).*

Proof. Note that $E = (\emptyset, \Omega, \emptyset, M_2^-)$ is a common state to each of the sequences π_1, π_4, and π_5. Starting from this common state, where machines M_1 and M_3 are free and the robot has just completed the loading of part $P_{\sigma(i)}$ onto M_2, the robot has three ways in which it can produce a single part and return to the same initial state. These consist of following one of the three sequences π_1, π_4, or π_5. The following results can be developed as in Section 6.3.1.

If π_1 or π_4 is used, we have an expression similar to that in the proof of Theorem 6.16, i.e.,

$$T^2_{1\sigma(i)\sigma(i+1)} = \alpha_1 + b_{\sigma(i)} + c_{\sigma(i)} + a_{\sigma(i+1)},$$

$$T^2_{4\sigma(i)\sigma(i+1)} = \alpha_4 + b_{\sigma(i)} + \max\{\beta_4, c_{4\sigma(i)}, a_{4\sigma(i+1)}\}.$$

Alternatively, if π_5 is used, we have

$$T^2_{5\sigma(i)\sigma(i+1)} = \alpha_5 + \max\{\beta_5 + c_{\sigma(i)}, b_{5\sigma(i)} + c_{\sigma(i)}, a_{5\sigma(i+1)}\},$$

where $\alpha_5 = 2\delta_2 + \epsilon_3 + \epsilon_4 - \eta, \beta_5 = 2\delta_1 + 2\delta_2 + 2\delta_3 + 2\delta_4 + \epsilon_1 + \epsilon_2 + \epsilon_5 + \epsilon_6 + \epsilon_7 + \epsilon_8 - 2\eta, a_{5\sigma(i)} = a_{\sigma(i)} + 2\delta_1 + \epsilon_1 + \epsilon_2$, and $b_{5\sigma(i)} = b_{\sigma(i)} + 2\delta_3 + 2\delta_4 + \epsilon_5 + \epsilon_6 + \epsilon_7 + \epsilon_8 - \eta$.

Thus, the cycle time expression becomes

$$T_\sigma = \sum_{i=1}^{n} h_{\sigma(i)\sigma(i+1)},$$

where $h_{\sigma(i)\sigma(i+1)} = \min\{T^2_{1\sigma(i)\sigma(i+1)}, T^2_{4\sigma(i)\sigma(i+1)}, T^2_{5\sigma(i)\sigma(i+1)}\}$. ∎

For $RF_3|(free,A,MP,\ RM(\pi_1,\ \pi_4, \pi_5))|\mu$, it is trivial to find the best robot move sequence for a given schedule σ of parts at the input hopper I. Each consecutive pair of parts $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ is processed using the quickest of the sequences π_1, π_4, and π_5.

To solve both of the above asymmetric traveling salesman problems, a computationally effective heuristic GENIUS, described by Gendreau et al. [65], was used. We now summarize the heuristic for $RF_3|(free,A,MP,$ *cyclic-n*$)|\mu$. It uses the solution procedures described earlier for all six 1-unit sequences, as well as heuristic solutions to the traveling salesman problems defined by the cycles in the proofs of Theorems 6.16 and 6.17. The smallest cycle time delivered by any of these is chosen. The heuristic thus solves the problem over robot move sequences that generate part scheduling problems which are polynomially solvable or have an effective heuristic. The steps of the heuristic are as follows.

Heuristic ThreeCell

Step 1: Use the algorithms in Section 6.3.2 to solve $RF_3|(free,A,MP,CRM\ (\pi_1))|\mu$ (resp., $RF_3|(free,A,MP,\ CRM(\pi_3))|\mu$, $RF_3|(free,A,\ MP,\ CRM(\pi_4))|\mu$, $RF_3|(free,A,MP,CRM(\pi_5))|\mu$). Let σ_1 (resp., σ_3, σ_4, σ_5) denote the associated part schedule and C_1 (resp., C_3, C_4, C_5) represent its cycle time.

Step 2: Use heuristic $MPS(\pi_2)$cycle for $RF_3|(free,A,MP,CRM(\pi_2))|\mu$, and heuristic $MPS(\pi_6)$cycle for $RF_3|(free,A,MP,CRM(\pi_6))|\mu$. Let σ_2 (resp., σ_6) denote the associated part schedule and C_2 (resp., C_6) represent its cycle time.

Step 3: Use GENIUS to solve $RF_3|(free,A,MP,RM(\pi_1,\pi_3,\pi_4))|\mu$ and $RF_3|(free,A,MP,RM(\pi_1,\pi_4,\pi_5))|\mu$. Let σ' (resp., σ'') denote the associated part schedule and C' (resp., C'') represent its cycle time.

Step 4: Compare the cycle times and select the best schedule. That is, find $C^* = \min\{C_1, C_2, C_3, C_4, C_5, C_6, C', C''\}$, and its associated robot move sequence and part schedule. Terminate.

To assess its performance, ThreeCell was tested on randomly generated problems. The instances were generated using the following four data sets:

data1: $\epsilon_i = 1.0, i = 1, \ldots, 8; \delta_i = 4.0, i = 1, \ldots, 4; \eta = 0.0$. The processing times are integers between 20 and 100 generated uniformly and independently.

data2: $\epsilon_i, \delta_i, \eta$ are the same as in data1, and the processing times are integers between 10 and 50 generated uniformly and independently.

data3: $\epsilon_i, \delta_i, \eta$ are the same as in data1, and the processing times are integers between 0 and 10 generated uniformly and independently.

data4: $\epsilon_i, \delta_i, \eta$ are the same as in data1, and the processing times are integers between 0 and 5 generated uniformly and independently.

The performance of ThreeCell was tested against a lower bound on the optimal cycle time computed as follows.

Procedure LBCT3$(a_i, b_i, c_i, i = 1, \ldots, n; \delta_1, \delta_2, \delta_3, \delta_4, \eta)$

Step 1: Call **MinCycle**$(a_i, b_i, i = 1, \ldots, n; \delta_1, \delta_2, \delta_3, \eta)$,
$LB_1 = C^*$,
Call **MinCycle**$(b_i, c_i, i = 1, \ldots, n; \delta_2, \delta_3, \delta_4, \eta)$,
$LB_2 = C^*$,
$LB_3 = \sum_{i=1}^{n} a_i + n(\epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 + 2\delta_1 + 2\delta_2 - \eta)$,

$$LB_4 = \sum_{i=1}^{n} b_i + n(\epsilon_3 + \epsilon_4 + \epsilon_5 + \epsilon_6 + 2\delta_2 + 2\delta_3 - \eta),$$
$$LB_5 = \sum_{i=1}^{n} c_i + n(\epsilon_5 + \epsilon_6 + \epsilon_7 + \epsilon_8 + 2\delta_3 + 2\delta_4 - \eta),$$
$$LB_6 = n\alpha_1 + \sum_{i=1}^{n} \min\{(a_i + b_i + c_i),\ (\min_{2\le j\le 3}\{\delta_j\} + \min\{a_i + b_i, a_i + c_i, b_i + c_i\}),\ \min_{2\le j\le 3}\{2\delta_j\} + \min\{a_i, b_i, c_i\}), \min_{2\le j\le 3}\{3\delta_j\})\}.$$

Step 2: $LB = \max\{LB_1, LB_2, LB_3, LB_4, LB_5, LB_6\}$.

In LBCT3, LB_1 (resp., LB_2) is the cycle time delivered by MinCycle(a_i, b_i, $i = 1, \ldots, n$; $\delta_1, \delta_2, \delta_3, \eta$) (resp., MinCycle($b_i$, c_i, $i = 1, \ldots, n$; δ_2, δ_3, δ_4, η)). LB_3, LB_4, and LB_5 are the processing times of parts at M_1, M_2, and M_3, respectively, plus the minimum robot activity times necessary to load a new part onto the machine (resp., M_1, M_2, or M_3) after completion of the previous part on the same machine. Similarly, LB_6 is another lower bound on the optimal cycle time (see Kamoun et al. [90] for a proof).

The results of testing ThreeCell against LBCT3 on 200 random instances generated using data1 (resp., data2, data3, data4) appear in Table 6.5 (resp., Tables 6.6, 6.7, 6.8). Much of the information there follows the format of Tables 6.3 and 6.4; *pmin* and *pmax* denote the lower and upper intervals of the range of processing times, respectively. In addition, column 7 (resp., 8, 9) shows O_6 (resp., O', O''), the number of instances for which the best cycle time found is given by C_6 (resp., C', C''). When the robot activity times are small compared to processing times, as is the case in data1 and data2, sequence π_6 dominates. However, when robot activity times are large compared to processing times, as in data3 and data4, the two MPS cycles composed of three 1-unit sequences provide better solutions. Heuristic solutions are typically closer to our lower bound in the latter case.

n	*pmin*	*pmax*	No.	MRPE	WRPE	O_6	O'	O''
5	20	100	50	4.258	11.264	50	0	0
10	20	100	50	8.647	14.167	50	0	0
20	20	100	50	10.616	16.726	50	0	0
50	20	100	50	12.651	17.040	50	0	0

Table 6.5. Performance of ThreeCell *vs.* LBCT3 Using data1.

n	$pmin$	$pmax$	No.	MRPE	WRPE	O_6	O'	O''
5	10	50	50	8.981	14.179	50	0	0
10	10	50	50	11.384	14.760	50	0	0
20	10	50	50	12.804	14.943	50	0	0
50	10	50	50	14.697	16.615	50	0	0

Table 6.6. Performance of ThreeCell *vs.* LBCT3 Using data2.

n	$pmin$	$pmax$	No.	MRPE	WRPE	O_6	O'	O''
5	0	10	50	2.568	5.998	0	25	25
10	0	10	50	2.763	5.158	0	24	26
20	0	10	50	2.752	4.137	0	26	24
50	0	10	50	3.152	3.998	0	28	22

Table 6.7. Performance of ThreeCell *vs.* LBCT3 Using data3.

n	$pmin$	$pmax$	No.	MRPE	WRPE	O_6	O'	O''
5	0	5	50	0.000	0.000	0	50	50
10	0	5	50	0.000	0.000	0	50	50
20	0	5	50	0.000	0.000	0	50	50
50	0	5	50	0.000	0.000	0	50	50

Table 6.8. Performance of ThreeCell *vs.* LBCT3 using data4.

6.8 Heuristics for Large Cells

There are exactly $m!$ potentially optimal 1-unit robot move sequences $\pi_{j,m}$, $j = 1, \ldots, m!$, in an m-machine cell. Section 6.6 classifies the MPS cycles formed by these CRM sequences according to the complexity of their associated part scheduling problems as follows:
Class U: Schedule independent (i.e., trivially solvable).
Class V: Problems that can be formulated as a TSP.
Class W: Unary NP-hard, but do not have a natural TSP structure (in the sense defined in Section 6.6.3).

We now illustrate how one can extend the ideas in heuristic ThreeCell to a four-machine cell. A similar extension is also possible for cells with more than four machines.

Given a W-cycle, the following heuristic usually finds a good approximate solution.

Heuristic Wcycle

Step 1: Set $j = 1$ and $m = 4$.

Step 2: Set the processing times of all the parts on M_j equal to zero. Consider the resulting cycle in a three-machine cell. The part scheduling problem can be solved heuristically, if it is either $CRM(\pi_{2,3})$ or $CRM(\pi_{6,3})$. Otherwise, it can be solved optimally in polynomial time. Let σ_j and C_j, respectively, denote the part schedule and the corresponding cycle time.

Step 3: Set $j = j + 1$. If $j \leq m$, then go to Step 2.

Step 4: Output the heuristic cycle time $C^* = \min_{1 \leq j \leq m}\{C_j\}$ and the corresponding part schedule, σ^*. Terminate.

We now identify the MPS cycles in a four-machine cell in which the part scheduling problem can be formulated as a TSP. These cycles are combinations of the 1-unit robot move sequences for which the part scheduling problem can be formulated as a TSP.

THEOREM 6.18 *Problem $RF_4|(free,A,MP,RM(\pi_1,\pi_4,\pi_{11},\pi_{14}))|\mu$ can be formulated as a TSP (in the sense defined in Section 6.6.3).*

Proof. Note that $E = (\emptyset, \emptyset, \emptyset, \Omega, M_4^-)$ is a state common to robot move sequences $\pi_{1,4}$, $\pi_{4,4}$, $\pi_{11,4}$, and $\pi_{14,4}$. Starting from E, where machines M_1, M_2, and M_3 are free and the robot has just completed the loading of part $P_{\sigma(i)}$ on M_4, the robot follows one of the four robot move sequences $\pi_{1,4}$, $\pi_{4,4}$, $\pi_{11,4}$, or $\pi_{14,4}$.

The expressions for the times $T^4_{1\sigma(i)\sigma(i+1)}$, $T^4_{4\sigma(i)\sigma(i+1)}$, $T^4_{11\sigma(i)\sigma(i+1)}$, and $T^4_{14\sigma(i)\sigma(i+1)}$ can be derived similarly to those in Section 6.3.1. Thus, the cycle time expression becomes

$$T_\sigma = \sum_{i=1}^{n} h_{\sigma(i)\sigma(i+1)},$$

where

$$h_{\sigma(i)\sigma(i+1)} = \min\{T^4_{1\sigma(i)\sigma(i+1)}, T^4_{4\sigma(i)\sigma(i+1)}, T^4_{11\sigma(i)\sigma(i+1)}, T^4_{14\sigma(i)\sigma(i+1)}\}$$

can be interpreted as the distance between city i to city $(i+1)$ in a TSP formulation. Two similar results now follow.

THEOREM 6.19 *Problem* $RF_4|(free,A,MP,RM(\pi_1,\pi_{13},\pi_{14},\pi_{17}))|\mu$ *can be formulated as a TSP, where* $h_{\sigma(i)\sigma(i+1)} = \min\{T^2_{1\sigma(i)\sigma(i+1)}, T^2_{13\sigma(i)\sigma(i+1)}, T^2_{14\sigma(i)\sigma(i+1)}, T^2_{17\sigma(i)\sigma(i+1)}\}$.

Proof. The proof is similar to that of Theorem 6.18.

THEOREM 6.20 *Problem* $RF_4|(free,A,MP,RM(\pi_1,\ \pi_9,\ \pi_{10},\ \pi_{11},\ \pi_{13},\ \pi_{14}))|\mu$ *can be formulated as a TSP, where* $h_{\sigma(i)\sigma(i+1)} = \min\{T^3_{1\sigma(i)\sigma(i+1)}, T^3_{9\sigma(i)\sigma(i+1)}, T^3_{10\sigma(i)\sigma(i+1)}, T^3_{11\sigma(i)\sigma(i+1)}, T^3_{13\sigma(i)\sigma(i+1)}, T^3_{14\sigma(i)\sigma(i+1)}\}$.

Proof. The proof is similar to that of Theorem 6.18.

REMARK 6.13 The part scheduling problems associated with any U- or V-cycle can be formulated as a TSP (in the sense defined in Section 6.6.3). All U- and V-cycles have an E^*_q state in which the robot has just completed the loading of a part onto machine M_q and all other machines are free. In a U-cycle, an E^*_q state occurs at each machine M_q, $1 \le q \le m$, but in any V-cycle, it occurs at one or more machines from $M_2, \ldots, M_m$. To formulate an MPS cycle with combined 1-unit robot move sequences as a TSP, each 1-unit sequence must have an E^*_q state on the same machine M_q. We are interested in identifying *maximal* feasible MPS cycles, i.e., those that allow as many combined 1-unit sequences as possible, since they offer the greatest number of scheduling options and thus dominate MPS cycles formed by CRM sequences associated with the individual 1-unit sequences. It is easy to see that there exists at least one V-cycle with an E^*_q state at any machine M_q, $2 \le q \le m$. The U-cycle can be combined with these V-cycles to create an MPS cycle with E^*_q at any machine M_q, $2 \le q \le m$. Since there are $m-1$ such machines, there are $m-1$ maximal combined cycles with a TSP formulation in an m-machine robotic cell.

We now summarize our heuristic for $RF_4|(free,A,MP,cyclic\text{-}n)|\mu$. It uses the solution procedures described earlier for all 24 1-unit sequences as well as heuristic solutions to the TSPs in Theorems 6.18, 6.19, and 6.20. The smallest cycle time is chosen. Thus, as before, we solve the problem over robot move sequences that generate part scheduling problems that are polynomially solvable or have an effective heuristic.

Heuristic FourCell

Step 1: Let C_1 denote the cycle time of any part schedule for the U-cycle.
Step 2: Use the Gilmore-Gomory algorithm to solve the part scheduling problems for all $V1$-cycles. Let C_2 denote the minimum cycle time.
Step 3: Use heuristic GENIUS by Gendreau et al. [65] to solve the part scheduling problems for all $V2$-cycles. Let C_3 denote the minimum cycle time.
Step 4: Use heuristic Wcycle to solve the part scheduling problems for all W-cycles. Let C_4 denote the minimum cycle time.
Step 5: Use heuristic GENIUS to solve the problems defined in Theorems 6.18, 6.19, and 6.20. Let C_5 denote the minimum cycle time.
Step 6: Find $C^* = \min\{C_1, C_2, C_3, C_4, C_5\}$, and its associated robot move sequence and part schedule. Terminate.

The performance of FourCell can be tested against a lower bound on the optimal cycle time computed using a procedure similar to LBCT3. The development of heuristic FourCell from heuristic ThreeCell may be seen as an induction step. The results of Section 6.6 provide a complete complexity mapping for cells of any size and an understanding of how robot move sequences proliferate when an extra machine is added to the cell. By using this understanding, as illustrated above, Heuristic FourCell can be extended to cells with any number of machines.

6.9 The Cell Design Problem

The cell design problem involves organizing several machines into cells which are arranged in a serial production process with intermediate buffers. This is a *Linked-Cell Manufacturing System*, or L-CMS. The major components of an L-CMS are the cells that are served either by human operators or by robots. In discussing an L-CMS, Black and Schroer [13] state: "The next generation of American factories (the factories with a future) will be designed with manufacturing and assembly cells linked together with a pull system for material and information control." The following example is taken from Miller and Walker [122]: "Massey Ferguson, the agricultural equipment manufacturer, installed a flexible automation system to link together eight stand-alone chuckers, shapers and shavers. The company chose Unimate robots after learn-

ing that special loaders (special purpose automation) would cost more in time and money than robots. ... The advantages of the system are at least 25% more productivity over the long term, as compared to a manual method, and flexibility – it can be changed over rapidly to accommodate four different sizes of pinion gears." In this example, Massey Ferguson used three robots to divide the eight machines into three cells, as in Figure 6.26.

We consider two cell design problems that underlie the robot move sequence and part scheduling issues considered earlier in this chapter. The solution procedures use algorithms discussed earlier in this chapter. In the first problem, we study how to divide the machines into cells that will operate efficiently. In the second problem, we allow the existence of finite capacity buffers, which may permit some rescheduling of parts between the cells. We then consider the problem of designing the smallest possible buffers that allow the cycle time for the entire manufacturing process to be minimum.

6.9.1 Forming Cells

Considered by itself, the cell formation problem is similar to the classical assembly line balancing problem (Hillier and Boling [84], Dar-El and Rubinovitch [43]), for which a variety of effective procedures are available. In the line balancing problem, a set of tasks with precedence relations are assigned to a certain number of workstations such that the maximum work load assigned to a workstation is minimized. In the cell formation problem, machines with linear precedence relations are assigned to cells so that the maximum cycle time of a cell is minimized, subject to the constraint that the number of machines assigned to a cell is bounded. This bound is equal to the maximum number of machines that the robot can serve in a cell. For a given cell design problem, the number of cells is fixed and is equal to the number of robots. The cell formation problem is much more complicated because the cycle time of a cell depends on robot move sequence and part scheduling issues that must be addressed simultaneously. Thus, it is unlikely that optimal solutions to the overall problem can be found for data sets of industrial size. This suggests the use of heuristic methods.

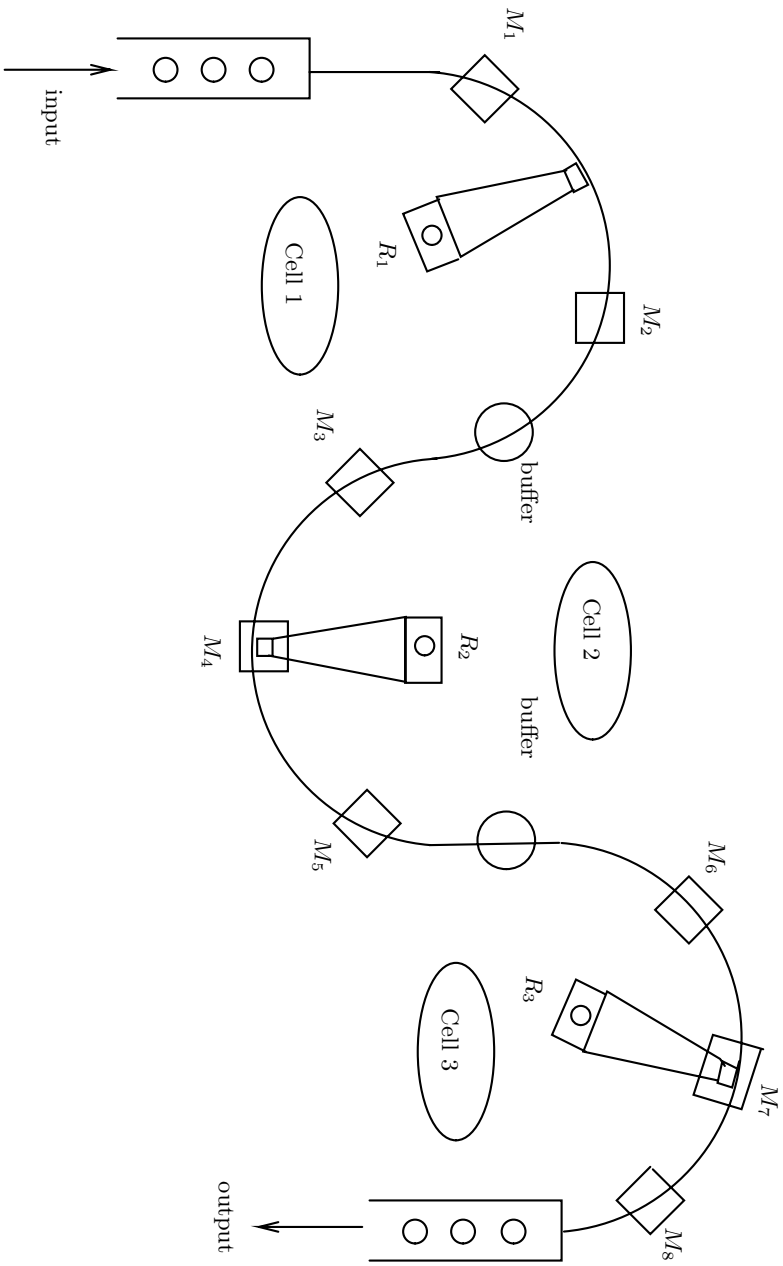


Figure 6.26. An Example of Partitioning Eight Machines into Three Robotic Cells.

To illustrate, we propose two such procedures that effectively partition the machines into two and three-machine cells. These procedures use Improved MinCycle (Section 6.2) and ThreeCell (Section 6.7.4) as subroutines. Let Γ denote the set of all possible partitions of the machines into two-machine and three-machine cells. For forming large cells, methods that schedule robotic cells with four or more machines (see Section 6.8) can also be used in the procedures below.

We now describe the heuristic CellDesign1 where the chosen partition ρ^*, the part schedule in each cell σ^*, and the robot move sequence in each cell are found by heuristically minimizing the cycle time of each cell individually. This procedure is appropriate when complete rescheduling of parts between the cells is possible. This may require large buffers between the cells.

Procedure CellDesign1
Step 1: $T^* = \infty$.
Step 2: For each partition $\rho \in \Gamma$, do
Let h_2 (resp., h_3) denote the number of RF_2 (resp., RF_3) cells in ρ.
Using **MinCycle**, solve the h_2 RF_2 cells for σ_i with cycle times $T_i, i = 1, \ldots, h_2$.
Using **ThreeCell**, solve the h_3 RF_3 cells for σ_i with cycle times T_i, i=$h_2 + 1, \ldots, h_2 + h_3$.
$\bar{T}$ =max$_{1 \le i \le h_2+h_3}\{T_i\}$.
If $\bar{T} < T^*$, then
$\sigma^* = (\sigma_1, \ldots, \sigma_{h_2+h_3})$.
$T^* = \bar{T}$.
$\rho^* = \rho$.
End If
End
Terminate.

Procedure CellDesign2, which follows, is suitable when no rescheduling of parts between the cells is possible. For each candidate partition and common part schedule, the best cycle time for each cell is obtained among all the robot move sequences for which the part scheduling problem is efficiently solvable (i.e., heuristic ThreeCell is used for $m = 3$; algorithm Improved Mincycle is used for $m = 2$). The schedule that

minimizes the largest cycle time of any cell, and thus minimizes the cycle time of the L-CMS, is a solution to the problem for a given partition. The partition chosen is the one that provides the minimum cycle time for the L-CMS. It is important to note that while CellDesign1 is used for a less constrained problem than CellDesign2, it may produce a larger cycle time since both procedures offer approximate solutions. The example in Section 6.9.3 illustrates this point.

Procedure CellDesign2
Step 1: $T^* = \infty$.
Step 2: For each partition $\rho \in \Gamma$, do
Let h_2 (resp., h_3) denote the number of RF_2 (resp., RF_3) cells in ρ.
Let h_3 denote the number of RF_3 cells in ρ.
Using **MinCycle**, solve the h_2 RF_2 cells for σ_i with cycle times $T_i, i = 1, \ldots, h_2$.
Using **ThreeCell**, solve the h_3 RF_3 cells for σ_i with cycle times $T_i, i = h_2 + 1, \ldots, h_2 + h_3$.
For $i = 1, \ldots, h_2 + h_3$, do
Solve all the cells using part schedule σ_i, giving cycle time $\hat{T}_j, j = 1, \ldots, h_2 + h_3$.
$\bar{T} = \max_{1 \le j \le h_2 + h_3} \{\hat{T}_j\}$.
If $\bar{T} < T^*$, then
$\sigma^* = \sigma_i$.
$T^* = \bar{T}$.
$\rho^* = \rho$.
End If
End
End

6.9.2 Buffer Design

If all the buffer capacities between the cells are large enough to reschedule and to feed cells without any starvation at the input of each cell, then the part scheduling problem is solved for each of the R cells by finding (locally) optimal schedules $\sigma_1, \ldots, \sigma_R$, and the corresponding robot move sequences.

We now consider the case where rescheduling of parts between the cells is not possible. If $\sigma_1 = \sigma_2 = \cdots = \sigma_R$, then this common schedule is optimal for the overall problem. Otherwise, CellDesign2 solves the overall problem R number of times by imposing on all the cells the schedule $\sigma_i, i = 1, \ldots, R$, and obtaining a best robot move sequence for each cell for each fixed σ_i. This procedure requires making tradeoffs in the part scheduling problem in one cell so as not to enter another cell with a part schedule that would generate a poor performance in that cell. The schedule σ^* found by CellDesign2 is a feasible solution to the problem where rescheduling is not allowed. To avoid unnecessary buffer build up, which is expensive in inventory holding cost, it is important to equalize the cycle time at all the cells. This can be achieved, for example, by increasing the values of the robot travel times in those cells that have a cycle time smaller than the maximum cycle time (T^*).

Even assuming that no rescheduling of parts between the cells is possible, a buffer may still be needed between the cells. This could happen if the cell data differ between two consecutive cells such that the interval between the instants at which two parts leave a cell is different from the interval between the instants at which they enter the next cell. We illustrate how by taking this into account, the required buffer sizes between the cells can be determined.

The following procedure finds the smallest buffer sizes between the cells that allow the cycle time for the overall problem to equal T^*. The basic idea of the procedure is to find the starting and completion times of parts by considering each cell independently and using the completion time of the first part P_1 in σ^* as a time reference (Step 1). Since the completion time of a part at one cell can be greater than the starting time of the same part in the next cell, we may need to delay the starting times of parts in later cells to avoid starvation at the cell input, at the expense of increasing the buffer size (Step 2). We form a set $fs = \{fs_1, \ldots, fs_{2n}\}$ that contains completion times and start times of all jobs in σ^* in the two adjacent cells q and $q+1$, respectively, for $1 \leq q \leq R-1$. First, we schedule the elements of fs in nondecreasing order. We then check each element in that order to see whether it is a completion time in cell q or a start time in cell $q+1$. If the entry is a completion time, we increase the buffer size by one unit, otherwise we decrease it by one unit. In the former case, we check if we have reached a new maximum value for the

buffer size (Step 3). We use the following notation:
$ft_{\sigma(i)}(q)$: the time the part scheduled in position i leaves cell q,
$st_{\sigma(i)}(q)$: the time the part scheduled in position i enters cell q,
$maxb_o(q)$: the maximum size of the buffer located between cells q and $q+1$,
b_o: buffer counter,
$T^0_{\sigma(i)\sigma(i+1)}$: the time between the pick up of part $P_{\sigma(i)}$ and the pick up of part $P_{\sigma(i+1)}$ at I, using a robot move sequence,
$T^{m+1}_{\sigma(i)\sigma(i+1)}$: the time between the drop off of part $P_{\sigma(i)}$ and the drop off of part $P_{\sigma(i+1)}$ at O, using a robot move sequence.

We will specify the robot move sequence when the cycle time expressions are developed for $T^0_{\sigma(i-1)\sigma(i)}$ and $T^{m+1}_{\sigma(i-1)\sigma(i)}$ for each cell. They are required for the following procedure and are derived below. The following procedure illustrates the determination of the buffer size requirement between cell q and cell $q+1$, for $q=1$. The procedure for $q=2,\ldots,R-1$, is similar.

Derivation of $T^0_{\sigma(i-1)\sigma(i)}$, $T^{m+1}_{\sigma(i-1)\sigma(i)}$ for Procedure Buffersize

Case 1. If cell q includes three machines, we have following subcases:

Case 1.1. If sequence $CRM(\pi_1)$ is used in cell q, then
$$T^{m+1}_{\sigma(i-1)\sigma(i)}(q) = +\alpha_1 + a_{\sigma(i)} + b_{\sigma(i)} + c_{\sigma(i)}, i=2,\ldots,n,$$
$$T^{0}_{\sigma(i-1)\sigma(i)}(q) = +\alpha_1 + a_{\sigma(i-1)} + b_{\sigma(i-1)} + c_{\sigma(i-1)}, i=2,\ldots,n.$$

Case 1.2. If sequence $CRM(\pi_2)$ is used in cell q, then
$$T^{m+1}_{\sigma(i-1)\sigma(i)}(q) = \alpha_2 + \beta_2 + w^i_2(q) + w^{i+1}_1(q) + w^i_3(q), i=2,\ldots,n,$$
$$T^{0}_{\sigma(i-1)\sigma(i)}(q) = \alpha_2 + \beta_2 + w^{i-2}_2(q) + w^{i-1}_1(q) + w^{i-2}_3(q), i=2,\ldots,n.$$

Case 1.3. If sequence $CRM(\pi_3)$ is used in cell q, then
$$T^{m+1}_{\sigma(i-1)\sigma(i)}(q) = \alpha_3 + \beta_3 + w^i_2(q) + a_{\sigma(i+1)} + w^i_3(q), i=2,\ldots,n,$$
$$T^{0}_{\sigma(i-1)\sigma(i)}(q) = \alpha_3 + \beta_3 + a_{\sigma(i-1)} + w^{i-2}_3(q) + w^{i-1}_2(q), i=2,\ldots,n.$$

Case 1.4. If sequence $CRM(\pi_4)$ is used in cell q, then
$$T^{m+1}_{\sigma(i-1)\sigma(i)}(q) = \alpha_4 + \beta_4 + w^i_1(q) + b_{\sigma(i)} + w^i_3(q), i=2,\ldots,n,$$
$$T^{0}_{\sigma(i-1)\sigma(i)}(q) = \alpha_4 + \beta_4 + w^{i-2}_3(q) + w^{i-1}_1(q) + b_{\sigma(i-1)}, i=2,\ldots,n.$$

Case 1.5. If sequence $CRM(\pi_5)$ is used in cell q, then
$$T^{m+1}_{\sigma(i-1)\sigma(i)}(q) = \alpha_5 + \beta_5 + w^i_1(q) + w^i_2(q) + c_{\sigma(i)}, i=2,\ldots,n,$$
$$T^{0}_{\sigma(i-1)\sigma(i)}(q) = \alpha_5 + \beta_5 + w^{i-2}_2(q) + c_{\sigma(i-2)} + w^{i-1}_1(q), i=2,\ldots,n.$$

Case 1.6. If sequence $CRM(\pi_6)$ is used in cell q, then
$$T^{m+1}_{\sigma(i-1)\sigma(i)}(q) = \alpha_6 + \beta_6 + w^i_2(q) + w^{i+1}_1(q) + w^i_3(q), i=2,\ldots,n,$$
$$T^{0}_{\sigma(i-1)\sigma(i)}(q) = \alpha_6 + \beta_6 + w^{i-3}_3(q) + w^{i-2}_2(q) + w^{i-1}_1(q), i=2,\ldots,n.$$

Case 1.7. If an MPS cycle with $RM(\pi_1,\pi_3,\pi_4)$ is used, then since analytical expressions are not available, use a deterministic simulation routine to find times $T^{m+1}_{\sigma(i-1)\sigma(i)}(q)$ and $T^{0}_{\sigma(i-1)\sigma(i)}(q)$.

Case 1.8. If an MPS cycle with $RM(\pi_1, \pi_4, \pi_5)$ is used, then since analytical expressions are not available, use a deterministic simulation routine to find times $T^{m+1}_{\sigma(i-1)\sigma(i)}(q)$ and $T^{0}_{\sigma(i-1)\sigma(i)}(q)$.

Case 2. If cell q includes two machines using $RM(\pi_1, \pi_2)$ (Section 6.2), we have the following subcases:

Case 2.1. $a'_{\sigma(i)} + b'_{\sigma(i-1)} \geq \mu$ and $a'_{\sigma(i+1)} + b'_{\sigma(i)} \geq \mu$
$T^{m+1}_{\sigma(i-1)\sigma(i)}(q) = \rho + \mu + w^i_1(q) + w^i_2(q), i = 2, \ldots, n.$

Case 2.2. $a'_{\sigma(i-1)} + b'_{\sigma(i-2)} \geq \mu$ and $a'_{\sigma(i)} + b'_{\sigma(i-1)} \geq \mu$
$T^{0}_{\sigma(i-1)\sigma(i)}(q) = \rho + \max\{\mu, b'_{\sigma(i-2)}, a'_{\sigma(i-1)}\}.$

Case 2.3. $a'_{\sigma(i)} + b'_{\sigma(i-1)} \leq \mu$ and $a'_{\sigma(i+1)} + b'_{\sigma(i)} \leq \mu$
$T^{m+1}_{\sigma(i-1)\sigma(i)}(q) = \rho + a'_{\sigma(i)} + b'_{\sigma(i)}, i = 2, \ldots, n.$

Case 2.4. $a'_{\sigma(i-1)} + b'_{\sigma(i-2)} \leq \mu$ and $a'_{\sigma(i)} + b'_{\sigma(i-1)} \leq \mu$
$T^{0}_{\sigma(i-1)\sigma(i)}(q) = \rho + a'_{\sigma(i-1)} + b'_{\sigma(i-1)}.$

Case 2.5. $a'_{\sigma(i)} + b'_{\sigma(i-1)} \geq \mu$ and $a'_{\sigma(i+1)} + b'_{\sigma(i)} \leq \mu$
$T^{m+1}_{\sigma(i-1)\sigma(i)}(q) = 2\delta_2 + 2\delta_3 + \epsilon_3 + \epsilon_4 + \epsilon_5 + \epsilon_6 - \eta + w^i_1(q) + b_{\sigma(i)},$
$i = 2, \ldots, n.$

Case 2.6. $a'_{\sigma(i-1)} + b'_{\sigma(i-2)} \geq \mu$ and $a'_{\sigma(i)} + b'_{\sigma(i-1)} \leq \mu$
$T^{0}_{\sigma(i-1)\sigma(i)}(q) = 2\delta_1 + 4\delta_2 + 4\delta_3 + \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 + 2\epsilon_5 + 2\epsilon_6$ -
$3\eta + w^i_1(q) + w^{i-1}_2(q) + b_{\sigma(i)}, i = 2, \ldots, n.$

Case 2.7. $a'_{\sigma(i)} + b'_{\sigma(i-1)} \leq \mu$ and $a'_{\sigma(i+1)} + b'_{\sigma(i)} \geq \mu$
$T^{m+1}_{\sigma(i-1)\sigma(i)}(q) = 4\delta_1 + 4\delta_2 + 2\delta_3 + 2\epsilon_1 + 2\epsilon_2 + \epsilon_3 + \epsilon_4 + \epsilon_5 + \epsilon_6.$
$-3\eta + w^i_2(q) + a_{\sigma(i)}, i = 2, \ldots, n$

Case 2.8. $a'_{\sigma(i-1)} + b'_{\sigma(i-2)} \leq \mu$ and $a'_{\sigma(i)} + b'_{\sigma(i-1)} \geq \mu$
$T^{0}_{\sigma(i-1)\sigma(i)}(q) = 2\delta_1 + 2\delta_2 + \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 - \eta + a_{\sigma(i)},$
$i = 2, \ldots, n.$

Procedure Buffersize

Step 1: For $q = 1$ do
$ft_{\sigma(1)}(q) = 0, st_{\sigma(1)}(q+1) = 0,$
$ft_{\sigma(i)}(q) = ft_{\sigma(i-1)}(q) + T^{m+1}_{\sigma(i-1)\sigma(i)},$
$st_{\sigma(i)}(q+1) = st_{\sigma(i-1)}(q+1) + T^{0}_{\sigma(i-1)\sigma(i)},$
If $ft_{\sigma(i)}(q) \leq st_{\sigma(i)}(q+1), i = 1, \ldots, n,$ then go to Step 3.
End

Step 2: If there exists a part i such that $ft_{\sigma(i)}(q) > st_{\sigma(i)}(q+1)$, then
Find $\max_{1 \leq i \leq n}\{ft_{\sigma(i)}(q) - st_{\sigma(i)}(q+1)\}$,
For $j = 1, \ldots, n$ do
$st_{\sigma(j)}(q+1) = st_{\sigma(j)}(q+1) + \max_{1 \leq i \leq n}\{ft_{\sigma(i)}(q) - st_{\sigma(i)}(q+1)\}.$
End
End If

Step 3: $ft = \{ft_{\sigma(1)}(q), \ldots, ft_{\sigma(n)}(q)\}$, $st = \{st_{\sigma(1)}(q+1), \ldots, st_{\sigma(n)}(q+1)\}$,
$fs_l = ft_{\sigma(l)}(q), l = 1, \ldots, n,$
$fs_{n+l} = st_{\sigma(l)}(q+1), l = 1, \ldots, n,$

Reindex such that $fs_1 \leq \ldots \leq fs_{2n}$, and $j = 1, i = 1, b_o = 0, maxb_o(r) = 0$,
For $l = 1, \ldots, 2n$, do
    If $fs_l = ft_{\sigma(j)}(q)$, then
        $b_o = b_o + 1, j = j + 1$.
    End If
    If $b_o > maxb_o(q)$, then
        $maxb_o(r) = b_o$.
    End If
    If $fs_l = st_{\sigma(i)}(q + 1)$, then
        $b_o = b_o - 1, i = i + 1$.
    End If
End

A similar procedure can be developed for CellDesign1, where a different schedule is used in each of the cells. The starting times and completion times are computed for each part-type in the same order in which the parts are produced in the cell. However, the starting times of parts in cell $q + 1$ are delayed by $\max_{1 \leq s \leq k, 1 \leq u \leq r_s} \{ft_{s,u}(q) - st_{s,u}(q + 1)\}$, where $ft_{s,u}(q)$ (resp., $st_{s,u}(q)$) denotes the time at which the uth part of type P_s enters (resp., leaves) cell q, $s = 1, \ldots, k$, $u = 1, \ldots, r_s$.

6.9.3 An Example

Using a randomly generated problem instance, we demonstrate how the above heuristics can be applied to obtain a good cell design. Consider the problem of designing an L-CMS that uses eight machines to process six part-types. The processing times and production ratios of these parts are given in Table 6.9. In this example, as described by Hall et al. [75], we assume that three robots are available to organize the machines into three cells in which the travel time between any two consecutive locations (of machines or buffers) equals δ, i.e., $\delta_i = \delta$. The data for the robots are as follows: $\epsilon_i = 1.0, \delta_i = 2.0, i = 1, \ldots, 8$, $\eta = 0.0$. We assume that the limited reach of the robots restricts the number of machines in any cell to be no more than three. Therefore, there are only three possible ways of partitioning the machines into three cells: $\rho_1 = (2, 3, 3), \rho_2 = (3, 2, 3)$, and $\rho_3 = (3, 3, 2)$, where $\rho_1 = (h, i, j)$ denotes a partition in which an h-machine cell is followed by an i-machine cell, and then by a j-machine cell.

Part-Type	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	r_i
P_1	10	3	2	4	2	18	35	37	12
P_2	50	18	38	8	30	16	27	19	1
P_3	35	16	9	39	29	23	20	13	26
P_4	33	19	10	14	10	29	20	19	1
P_5	1	8	2	23	31	35	17	15	7
P_6	5	20	6	30	21	14	7	11	4

Table 6.9. An Instance of the Cell Design Problem with $m = 8$ and $k = 6$.

We use algorithms Improved Mincycle and ThreeCell to find cycle times for each cell, assuming that we allow complete rescheduling between the cells. That is, there exists between each consecutive pair of cells a buffer of capacity large enough to feed the cells without starvation. The solution delivered by CellDesign1 is ρ_2 in Table 6.10 and has a cycle time of 2083, which represents an improvement of 7.1% and 6.7% over the cycle times of partitions ρ_1 and ρ_3, respectively. The solution delivered by CellDesign2 is an optimal solution to the above cell design problem, and it has the same partition, ρ_2, in Table 6.11. However, the cycle time has decreased to 2082 because of the change in the schedule of parts in Cell 1, which is no longer the bottleneck cell. The improvement of the cycle time in this case over other partitions or schedules ranges from 1.4% to 14.8%. To obtain the appropriate buffer design, it is essential to equalize the cycle times of Cells 1 and 3 to 2082. This can be achieved by using $\delta_1 = 2.1667$ in Cell 1, $\delta_1 = 2.5527$ in Cell 3, and keeping the values of all other δ_i unchanged. The necessary buffer capacities required between Cells 1 and 2 and between Cells 2 and 3 delivered by Buffersize are 7 parts for both buffers.

6.9.4 Computational Testing

Procedures CellDesign1 and CellDesign2 were tested on randomly generated problems. The performance of these procedures was compared against a lower bound on the optimal cycle time which was found using the following procedure. Recall that a_i (resp., b_i, c_i) denotes the processing time of part i on machine M_1 (resp., M_2, M_3) for the cell design problem, $i = 1, \ldots, n$.

Cell Partition	Cell	Sequence used	Cycle time
	1	π_1, π_2	1859.0
ρ_1=(2,3,3)	2	π_6	2230.0
	3	π_6	2043.0
	1	π_6	2083.0
ρ_2=(3,2,3)	2	π_1, π_2	2082.0
	3	π_6	2043.0
	1	π_6	2083.0
ρ_3=(3,3,2)	2	π_6	2222.0
	3	π_1, π_2	1796.0

Table 6.10. Results for CellDesign1 on the Example in Table 6.9.

Procedure LBCD$(a_i, b_i, c_i, i = 1, \ldots, n, \delta_1, \delta_2, \delta_3, \delta_4, \eta)$
Step 1: For each partition $\rho \in \Gamma$, do
For $r = 1, \ldots,$ R, do
If cell r is a two-machine cell, then
Call **MinCycle**$(a_i, b_i, i = 1, \ldots, n, \delta_1, \delta_2, \delta_3, \eta)$,
$LB(r) = C^*$.
End If
If cell r is a three-machine cell, then
Call **LBCT3**$(a_i, b_i, c_i, i = 1, \ldots, n, \delta_1, \delta_2, \delta_3, \delta_4, \eta)$,
$LB(r) = LB$.
End If
End
$LB_\rho = \max_{1 \le r \le R}\{LB(r)\}$.
End
Step 2: $LB^{'} = \min_{\rho \in \Gamma}\{LB_\rho\}$.

The results of testing CellDesign1 (resp., CellDesign2) on 10 randomly generated problem instances each from data1, data2, data3, and data4 appear in Table 6.12 (resp., Table 6.13). As in Section 6.9.3, we have eight machines that are being divided into three cells. Much of the information follows the format of Tables 6.3 and 6.4. In addition, in column 5 (resp., column 6) we provide MRPI (resp., HRPI), the mean relative percentage improvement (resp., the highest relative percentage improvement), computed over the 10 instances, relative to the cycle time

of the worst partition and schedule considered in the procedure. Since the errors shown in Tables 6.12 and 6.13 are computed relative to a lower bound, the actual heuristic performance relative to the optimal cycle time may be better than that suggested by the values shown here. The schedules obtained in CellDesign1 and CellDesign2 may be improved further by employing a neighborhood search.

We conclude this chapter with a brief discussion of the results for multiple-part scheduling problems in no-wait robotic cells. For $m = 2$

Cell Partition	Optimized Cell	Cell	Sequence used	Cycle time
ρ_1=(2,3,3)	1	1	π_1, π_2	1859.0
		2	π_6	2390.0
		3	π_6	2083.0
ρ_1=(2,3,3)	2	1	π_1, π_2	1922.0
		2	π_6	2230.0
		3	π_6	2040.0
ρ_1=(2,3,3)	3	1	π_1, π_2	1913.0
		2	π_6	2249.0
		3	π_6	2043.0
ρ_2=(3,2,3)	1	1	π_6	2083.0
		2	π_1, π_2	2134.0
		3	π_6	2032.0
ρ_2=(3,2,3)	2	1	π_6	2065.0
		2	π_1, π_2	2082.0
		3	π_6	2040.0
ρ_2=(3,2,3)	3	1	π_6	2068.0
		2	π_1, π_2	2111.0
		3	π_6	2043.0
ρ_3=(3,3,2)	1	1	π_6	2083.0
		2	π_6	2252.0
		3	π_1, π_2	1812.0
ρ_3=(3,3,2)	2	1	π_6	2065.0
		2	π_6	2222.0
		3	π_1, π_2	1802.0
ρ_3=(3,3,2)	3	1	π_6	2065.0
		2	π_6	2252.0
		3	π_1, π_2	1796.0

Table 6.11. Results for CellDesign2 on the Example in Table 6.9.

n	Data Set	No.	MRPE	WRPE	MRPI	HRPI
50	data1	10	13.551	16.489	1.583	3.996
50	data2	10	15.381	17.115	1.314	3.056
50	data3	10	3.409	4.383	0.217	1.038
50	data4	10	0.000	0.000	0.581	1.266

Table 6.12. Performance of CellDesign1 *vs.* LBCD.

n	Data Set	No.	MRPE	WRPE	MRPI	HRPI
50	data1	10	16.470	18.763	5.695	9.111
50	data2	10	16.045	16.961	2.925	3.769
50	data3	10	3.660	4.541	1.645	2.672
50	data4	10	0.000	0.000	0.581	1.266

Table 6.13. Performance of CellDesign2 *vs.* LBCD.

(problem $RF_2|(no\text{-}wait,E,MP,CRM)|\mu$), the part scheduling problem in an MPS cycle with $CRM(\pi_2)$ can be solved using the Gilmore-Gomory algorithm (Agnetis [2]). For $m = 3$, the part scheduling problem in an MPS cycle with $CRM(\pi_1)$ is trivial. The part scheduling problems in MPS cycles with $CRM(\pi_3)$, $CRM(\pi_4)$, and $CRM(\pi_5)$ can be solved by the Gilmore-Gomory algorithm, and those with $CRM(\pi_2)$ and $CRM(\pi_6)$ are strongly NP-hard (Agnetis and Pacciarelli [3]). These are further elaborated in Chapter 9.

Chapter 7

MULTIPLE-PART-TYPE PRODUCTION: DUAL-GRIPPER ROBOTS

We examine robotic cells with dual-gripper robots that process lots containing different types of parts. We use notation and many of the concepts defined in Chapters 4 and 6. As in Chapter 4, we consider additive travel-time cells that are circular with the input (I) and output (O) buffers at a common location. We focus on obtaining an optimal MPS cycle within a class of robot move sequences, called CRM sequences (Chapter 6). In Section 7.1, we show how to derive, for a given part schedule, the cycle time expressions for CRM sequences. We show that for a two-machine robotic cell there exist 13 potentially optimal robot move sequences. Section 7.2 is devoted to the complexity status of the problem. We prove that the general problem of multiple-part scheduling in a two-machine robotic cell served by a dual-gripper robot is strongly NP-hard. In particular, we demonstrate that the recognition versions of the part scheduling problem for six of the 13 potentially optimal robot move sequences are strongly NP-complete. For the remaining seven sequences, an optimal part schedule can be found in polynomial time. In Section 7.3, we consider a special case of the problem for which we identify a robot move sequence that delivers the minimal MPS cycle time. This variant arises often in practice and reflects possibly the most common case in real-world robotic systems. We develop a heuristic for the six NP-hard problems in Section 7.4. The worst-case performance bound of the heuristic is established in Section 7.4.1. In Section 7.5, we propose a heuristic to solve the general problem in a two-machine robotic

cell and discuss some computational experience. In Section 7.6, we do a comparative computational study to quantify the productivity improvement effected by a dual-gripper cell over a single-gripper cell. Section 7.7 solves the general m-machine problem by extending the heuristic for the two-machine problem and evaluates its performance.

As in Chapter 4, the times associated with the robot movements are as follows:

δ: The time taken by the robot to travel between two consecutive machines M_{j-1} and M_j, $1 \leq j \leq m+1$. The robot travel time between locations x and y via the shortest route is denoted by $\ell(x, y)$. Thus, the travel-time of the robot from M_i to M_j is $\ell(M_i, M_j) = \ell(M_j, M_i) = \delta \times \min\{|i-j|, m+1-|i-j|\}$. For example, in a seven-machine cell, $\ell(I/O, M_1) = \delta$, $\ell(M_1, M_3) = 2\delta$, $\ell(M_1, M_7) = 2\delta$, $\ell(M_3, I/O) = 3\delta$, $\ell(M_5, M_2) = 3\delta$, etc.

ϵ: The time taken by the robot to pick up/drop off a part at I/O or the time taken by the robot to load/unload a part onto/from a machine.

θ: The gripper switching time. We assume that the time required for the dual-gripper robot to reposition its grippers while traveling from one machine to another (i.e., when there are two successive operations executed on different machines and requiring different grippers) does not exceed travel time of the robot between the machines. To put it formally, let $\bar{\theta}$ be the time taken by a robot to switch its grippers while traveling between machines. For all the problems considered in this chapter, we assume that $\bar{\theta} \leq \delta$. As in Chapter 4, the adequateness of this assumption is governed primarily by its practical relevance.

We first focus on the problem of scheduling multiple part-types in a two-machine robotic cell ($m = 2$). The objective of scheduling the robotic cell under consideration is to find an MPS cycle with the minimum cycle time. This requires a simultaneous determination of a part schedule and a move sequence. In this chapter we focus on CRM sequences. We have two reasons to consider CRM sequences: first, they are the easiest to implement and control, and second, (as will be seen later) efficient cyclic solutions can be found for the part scheduling problem under CRM sequences.

We now present the notation used for our analysis of two-machine cells:

$\mathcal{P} = \{P_1, \ldots, P_n\}$: a set of parts to be processed.

a_k, b_k: the processing times of part P_k on machines M_1 and M_2, respectively. For cells with three or more machines, the processing time of part P_k on machine M_j is denoted by $p_{k,j}$.

$\sigma = (\sigma(1), \ldots, \sigma(n))$: the schedule of parts of an MPS, with $\sigma(i) = k$ denoting that the ith part in the schedule σ is P_k.

$a_{\sigma(j)}$ (resp., $b_{\sigma(j)}$) : the processing time of the jth part on machine M_1 (resp., M_2) in schedule σ.

$R_i^-(j,0)$ (or $R_i^-(0,j)$), $i = 1,2,3$, and $j = 0,1,2,3$: the state in which the robot has just finished loading a part onto M_i and has a part that requires processing next on some machine M_j; $j = 0$ (state $R_i^-(0,0)$) denotes that the robot does not hold any part.

$R_i^+(j,k)$, $i = 1,2,3$, $j = 0,1,2,3$, and $k = 0,1,2,3$: the state in which the robot has just finished unloading a part from M_i, and now holds parts that require processing next on machines M_j and M_k, respectively; $j = 0$ (resp., $k = 0$) denotes that the robot has only a part intended for M_k (resp., M_j). Note that either $j = i+1$ or $k = i+1$, and that $j = k = 0$ cannot occur.

$C_{u,v}$: vth 1-unit cycle under Case u, $u = 1,2,3,4$ (see Chapter 4).

$S_{u,v}$: The CRM sequence constructed from $C_{u,v}$ (See Section 7.1).

$T_{u,v}^{\sigma}$: The cycle time of an MPS cycle under the CRM sequence $S_{u,v}$ subject to a part schedule $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$.

$T_{u,v}^{\sigma} = T_{u,v}^{h\sigma(1)\sigma(2)} + T_{u,v}^{h\sigma(2)\sigma(3)} + \ldots + T_{u,v}^{h\sigma(n)\sigma(1)}$, where $T_{u,v}^{h\sigma(i)\sigma(i+1)}$ is the elapsed time between the following two successive events in the MPS cycle under the CRM sequence $S_{u,v}$ and the part schedule σ: (i) the robot has just completed unloading part $P_{\sigma(i)}$ from M_h, (ii) the robot has just completed unloading part $P_{\sigma(i+1)}$ from M_h, $0 \le h \le m+1$.

$w_j^i = w_j^i(u,v;\sigma)$: The waiting time of the robot at machine M_j before unloading part $P_{\sigma(i)}$ from M_j for the MPS cycle corresponding to

$S_{u,v}$ and σ. Our notation suppresses the dependence of w_j^i on $S_{u,v}$ and σ, since this dependence will be clear from the context.

$T_{u,v}^*$: The optimal value of the cycle time $T_{u,v}^\sigma$ over the set of all possible part schedules σ. Thus, for any schedule $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$, we have $T_{u,v}^\sigma \geq T_{u,v}^*$.

$\sigma_{u,v}^*$: A permutation that delivers the optimal value of $T_{u,v}^*$, i.e., $T_{u,v}^* = T_{u,v}^{\sigma_{u,v}^*}$.

7.1 Two-Machine Cells: Undominated CRM Sequences

We limit our search for a solution of $RF_2^2|(free,A,MP,cyclic\text{-}n)|\mu$ to CRM sequences. To achieve this, we identify undominated CRM sequences by showing that it is sufficient to consider only these cycles in our search for optimality. To define CRM sequences more precisely, we need to specify all 1-unit cycles defined in terms of robot move sequences. Note that 1-unit cycles are defined in the context of single-part-type production.

In our approach to construct CRM sequences, we are concerned with robot move sequences associated with 1-unit cycles. A 1-unit cycle can be specified by a sequence of the robotic cell states as described in Chapter 4, e.g., the cycle $C_{3,10} = \{R_2^+(3,2),\ R_2^-(3,0),\ R_3^-(0,0),\ R_3^+(1,0),\ R_1^+(1,2),\ R_1^-(0,2)\}$. Since this sequence of states uniquely defines the required robot move sequence, we shall also use the notation $C_{3,10}$ to denote the robot move sequence associated with the cycle. We denote by $S_{u,v}$ the CRM sequence obtained from the 1-unit robot move sequence $C_{u,v}$. Recall that a CRM sequence is a concatenation of n identical 1-unit robot move sequences. That is, $S_{u,v}$ is a robot move sequence in which the sequence $C_{u,v}$ is repeated exactly n times in succession. For example, when $n = 3$, we can write

$$S_{3,10} = \{C_{3,10}, C_{3,10}, C_{3,10}\},\ \text{or}$$

$$\begin{aligned} S_{3,10} = \{&R_2^+(3,2), R_2^-(3,0), R_3^-(0,0), R_3^+(1,0), R_1^+(1,2), R_1^-(0,2),\\ &R_2^+(3,2), R_2^-(3,0), R_3^-(0,0), R_3^+(1,0), R_1^+(1,2), R_1^-(0,2),\\ &R_2^+(3,2), R_2^-(3,0), R_3^-(0,0), R_3^+(1,0), R_1^+(1,2), R_1^-(0,2)\}. \end{aligned}$$

Note that for MPS production in a two-machine robotic cell, there are 52 possible CRM sequences corresponding to the 52 1-unit robot move cycles described in Chapter 4. Let $\mathcal{S}_0 = \{S_{u,v},$ for all u and $v\}$ denote the set of all 52 CRM sequences. Our first task is to demonstrate that in the search for an optimal solution to $RF_2^2|(free,A,MP,S_{u,v} \in \mathcal{S}_0)|\mu$, it is possible to eliminate all but 13 of the 52 CRM sequences. For each of the 52 sequences, we first provide an expression to evaluate the cycle time subject to a given part schedule σ.

The purpose here is to find an optimal solution, consisting of a CRM sequence S_{u^*,v^*} from $\mathcal{S}_0$ and a part schedule σ^*, that has the minimum cycle time. That is, $T^{\sigma^*}_{u^*,v^*} \leq T^{\sigma}_{u,v}$ for each σ and each $S_{u,v} \in \mathcal{S}_0$. We now illustrate the derivation of the cycle time $T^{\sigma}_{3,10}$. As the production of MPS parts is cyclic, we define

$$\begin{aligned} a_{\sigma(n+i)} &= a_{\sigma(i)},\ b_{\sigma(n+i)} = b_{\sigma(i)}, \\ w_j^{n+i} &= w_j^i \text{ for } i = 1, \ldots, n,\ j = 1, 2. \end{aligned} \tag{7.1}$$

LEMMA 7.1 *In $RF_2^2|(free,A,MP,S_{3,10})|\mu$, the cycle time of an MPS cycle for a schedule σ is*

$$T^{\sigma}_{3,10} = 2n\epsilon + n\theta + \sum_{i=1}^{n} \max\{a_{\sigma(i+1)} - w_2^{i-1}, b_{\sigma(i)}, \alpha\}, \tag{7.2}$$

where

$$w_2^i = \max\{0, b_{\sigma(i)} - \max\{\alpha, a_{\sigma(i+1)} - w_2^{i-1}\}\} \tag{7.3}$$

and $\alpha = 3\delta + 4\epsilon + \theta$.

Proof. Recall that the cycle time $T^{\sigma}_{u,v}$ can be expressed as

$$T^{\sigma}_{u,v} = T^{h\sigma(1)\sigma(2)}_{u,v} + T^{h\sigma(2)\sigma(3)}_{u,v} + \ldots + T^{h\sigma(n)\sigma(1)}_{u,v},$$

where the expression for $T^{h\sigma(i)\sigma(i+1)}_{u,v}$, the time elapsed between the unloading of part $P_{\sigma(i)}$ and the next unloading of part $P_{\sigma(i+1)}$ at M_h, can be derived by adding the robot activity times between those two events in an MPS cycle corresponding to $S_{u,v}$ and σ. Below, we demonstrate this procedure for $T^{2\sigma(i)\sigma(i+1)}_{3,10}$.

Begin in the state $R_2^+(3,2)$: the robot has just unloaded part $P_{\sigma(i)}$ from M_2 and holds another part $P_{\sigma(i+1)}$ to be processed on M_2; part $P_{\sigma(i+2)}$ is being processed at M_1. The robot switches to the other gripper

(θ), loads part $P_{\sigma(i+1)}$ at M_2 (ϵ), moves to I/O (δ), drops off $P_{\sigma(i)}$ at I/O (ϵ), picks up $P_{\sigma(i+3)}$ at I/O (ϵ), moves to M_1 (δ), waits for $P_{\sigma(i+2)}$ to be finished at M_1 (w_1^{i+2}), unloads $P_{\sigma(i+2)}$ from M_1 (ϵ), switches to the other gripper (θ), loads $P_{\sigma(i+3)}$ at M_1 (ϵ), moves to M_2 (δ), waits for $P_{\sigma(i+1)}$ to be finished at M_2 (w_2^{i+1}), and unloads $P_{\sigma(i+1)}$ from M_2 (ϵ). Thus,

$$T_{3,10}^{2\sigma(i)\sigma(i+1)} = 3\delta + 6\epsilon + 2\theta + w_1^{i+2} + w_2^{i+1},$$

where

$$\begin{aligned} w_1^{i+2} &= \max\{0, a_{\sigma(i+2)} - 3\delta - 4\epsilon - \theta - w_2^i\}, \\ w_2^{i+1} &= \max\{0, b_{\sigma(i+1)} - 3\delta - 4\epsilon - \theta - w_1^{i+2}\}. \end{aligned}$$

By simplifying the above two expressions for waiting times, we obtain

$$\begin{aligned} w_1^{i+1} &= \max\{0, a_{\sigma(i+1)} - \max\{\alpha, b_{\sigma(i-1)} - w_1^i\}\}, \\ w_2^i &= \max\{0, b_{\sigma(i)} - \max\{\alpha, a_{\sigma(i+1)} - w_2^{i-1}\}\}. \end{aligned}$$

By substituting $w_1^{i+2} + w_2^{i+1}$ into the expression for $T_{3,10}^{2\sigma(i)\sigma(i+1)}$ and by using the steady-state conditions (7.1), the required expression for the cycle time $T_{3,10}^{\sigma} = \sum_{i=1}^{n} T_{3,10}^{2\sigma(i)\sigma(i+1)}$ can be derived. ∎

Following a similar procedure, we derive the cycle time expressions for all CRM sequences corresponding to Cases 1–3 (see Chapter 4), and present them in Tables 7.1–7.3. As discussed in Chapter 4, Case 4 need not be considered because the two cycles belonging to this case are equivalent to two cycles in Case 1. A CRM sequence S_{u_1,v_1} is said to be dominated by another CRM sequence S_{u_2,v_2}, if $T_{u_1,v_1}^{\sigma} \geq T_{u_2,v_2}^{\sigma}$ for each part schedule σ. The last column in each of Tables 7.1–7.3 presents the dominance relations. We note that there are only 13 undominated CRM sequences (labeled as UD). Thus, in our search for optimality, we may restrict our analysis to only the set $\mathcal{S}$ of these 13 CRM sequences. Thus,

$$\begin{aligned} \mathcal{S} = \{&S_{1,1}, S_{1,4}, S_{1,13}, S_{1,14}, S_{1,15}, S_{3,3}, S_{3,4}, S_{3,5}, S_{3,6}, S_{3,9}, \\ &S_{3,10}, S_{3,11}, S_{3,16}\}. \end{aligned}$$

CRM	**Cycle Time**	**Dominance**
$S_{1,1}$	$T^{\sigma}_{1,1} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{1,1} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon, a_{\sigma(i+1)} + 3\delta + 5\epsilon, b_{\sigma(i)} + 2\delta + 3\epsilon\}$	UD
$S_{1,2}$	$T^{\sigma}_{1,2} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{1,2} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon + \theta, a_{\sigma(i+1)} + 3\delta + 5\epsilon + \theta, b_{\sigma(i)} + 2\delta + 3\epsilon\}$	$T^{\sigma}_{1,2} \geq T^{\sigma}_{1,1}$
$S_{1,3}$	$T^{\sigma}_{1,3} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{1,3} = \sum_{i=1}^{n} \max\{6\delta + 6\epsilon, a_{\sigma(i+1)} + 3\delta + 4\epsilon, b_{\sigma(i)} + 2\delta + 3\epsilon\}$	$T^{\sigma}_{1,3} \geq T^{\sigma}_{3,5}$
$S_{1,4}$	$T^{\sigma}_{1,4} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{1,4} = \sum_{i=1}^{n} \max\{a_{\sigma(i+1)} + 2\delta + 4\epsilon, b_{\sigma(i)} + 4\delta + 6\epsilon\}$	UD
$S_{1,5}$	$T^{\sigma}_{1,5} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{1,5} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon, a_{\sigma(i+1)} + 3\delta + 5\epsilon, b_{\sigma(i)} + 3\delta + 5\epsilon\}$	$T^{\sigma}_{1,5} \geq T^{\sigma}_{1,1}$
$S_{1,6}$	$T^{\sigma}_{1,6} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{1,6} = \sum_{i=1}^{n} \max\{6\delta + 6\epsilon, a_{\sigma(i+1)} + 4\delta + 5\epsilon, b_{\sigma(i)} + 3\delta + 4\epsilon\}$	$T^{\sigma}_{1,6} \geq T^{\sigma}_{1,1}$
$S_{1,7}$	$T^{\sigma}_{1,7} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{1,7} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon, a_{\sigma(i+1)} + 3\delta + 5\epsilon\} + \sum_{i=1}^{n} b_i$	$T^{\sigma}_{1,7} \geq T^{\sigma}_{1,14}$
$S_{1,8}$	$T^{\sigma}_{1,8} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{1,8} = \sum_{i=1}^{n} (4\delta + 6\epsilon + a_{\sigma(i+1)} + b_{\sigma(i+1)})$	$T^{\sigma}_{1,8} \geq T^{\sigma}_{1,13}$
$S_{1,9}$	$T^{\sigma}_{1,9} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{1,9} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon, b_{\sigma(i+1)} + 3\delta + 5\epsilon\} + \sum_{i=1}^{n} a_i$	$T^{\sigma}_{1,9} \geq T^{\sigma}_{1,1}$
$S_{1,10}$	$T^{\sigma}_{1,10} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{1,10} = \sum_{i=1}^{n} \max\{a_{\sigma(i+1)} + 2\epsilon + \theta, b_{\sigma(i)} + 3\delta + 6\epsilon + 2\theta\}$	$T^{\sigma}_{1,10} \geq T^{\sigma}_{1,14}$
$S_{1,11}$	$T^{\sigma}_{1,11} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{1,11} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon + \theta, a_{\sigma(i+1)} + 2\delta + 3\epsilon, b_{\sigma(i)} + 3\delta + 5\epsilon + \theta\}$	$T^{\sigma}_{1,11} \geq T^{\sigma}_{1,15}$
$S_{1,12}$	$T^{\sigma}_{1,12} = T^{\sigma}_{1,13} + n\theta$	$T^{\sigma}_{1,12} \geq T^{\sigma}_{1,13}$
$S_{1,13}$	$T^{\sigma}_{1,13} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{1,13} = \sum_{i=1}^{n} (3\delta + 6\epsilon + a_{\sigma(i+1)} + b_{\sigma(i+1)})$	UD
$S_{1,14}$	$T^{\sigma}_{1,14} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{1,14} = \sum_{i=1}^{n} \max\{a_{\sigma(i+1)} + 2\epsilon + \theta, b_{\sigma(i)} + 3\delta + 6\epsilon + \theta\}$	UD
$S_{1,15}$	$T^{\sigma}_{1,15} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{1,15} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon, a_{\sigma(i+1)} + 2\delta + 3\epsilon, b_{\sigma(i)} + 3\delta + 5\epsilon\}$	UD
$S_{1,16}$	$T^{\sigma}_{1,16} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{1,16} = \sum_{i=1}^{n} \max\{a_{\sigma(i+1)} + 2\delta + 3\epsilon, b_{\sigma(i)} + 5\delta + 6\epsilon\}$	$T^{\sigma}_{1,16} \geq T^{\sigma}_{1,15}$
$S_{1,17}$	$T^{\sigma}_{1,17} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{1,17} = \sum_{i=1}^{n} \max\{6\delta + 6\epsilon, a_{\sigma(i+1)} + 3\delta + 4\epsilon, b_{\sigma(i)} + 4\delta + 5\epsilon\}$	$T^{\sigma}_{1,17} \geq T^{\sigma}_{1,18}$
$S_{1,18}$	$T^{\sigma}_{1,18} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{1,18} = \sum_{i=1}^{n} \max\{6\delta + 6\epsilon, a_{\sigma(i+1)} + 3\delta + 4\epsilon, b_{\sigma(i)} + 3\delta + 4\epsilon\}$	$T^{\sigma}_{1,18} \geq T^{\sigma}_{1,3}$

Table 7.1. Case 1: Cycle-Time Expressions for Scheduling Multiple-Part-Type Production Using CRM Sequences.

CRM	Cycle Time	Dominance
$S_{2,1}$	$T^{\sigma}_{2,1} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{2,1} = \sum_{i=1}^{n}(3\delta + 6\epsilon + a_{\sigma(i+1)} + b_{\sigma(i+1)}) = T^{\sigma}_{1,13}$	$T^{\sigma}_{2,1} \geq T^{\sigma}_{1,13}$
$S_{2,2}$	$T^{\sigma}_{2,2} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{2,2} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon, b_{\sigma(i+1)} + 3\delta + 5\epsilon\} + \sum_{i=1}^{n} a_i$	$T^{\sigma}_{2,2} \geq T^{\sigma}_{2,3}$
$S_{2,3}$	$T^{\sigma}_{2,3} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{2,3} = \sum_{i=1}^{n} \max\{4\delta + 6\epsilon, b_{\sigma(i+1)} + 2\delta + 4\epsilon\} + \sum_{i=1}^{n} a_i$	$T^{\sigma}_{2,3} \geq T^{\sigma}_{3,3}$
$S_{2,4}$	$T^{\sigma}_{2,4} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{2,4} = \sum_{i=1}^{n} \max\{4\delta + 6\epsilon + \theta, b_{\sigma(i+1)} + 2\delta + 4\epsilon\} + \sum_{i=1}^{n} a_i$	$T^{\sigma}_{2,4} \geq T^{\sigma}_{2,3}$
$S_{2,5}$	$T^{\sigma}_{2,5} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{2,5} = \sum_{i=1}^{n} \max\{4\delta + 6\epsilon, a_{\sigma(i+1)} + 2\delta + 4\epsilon\} + \sum_{i=1}^{n} b_i$	$T^{\sigma}_{2,5} \geq T^{\sigma}_{1,4}$
$S_{2,6}$	$T^{\sigma}_{2,6} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{2,6} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon, a_{\sigma(i+1)} + 3\delta + 5\epsilon\} + \sum_{i=1}^{n} b_i$	$T^{\sigma}_{2,6} \geq T^{\sigma}_{2,5}$
$S_{2,7}$	$T^{\sigma}_{2,7} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{2,7} = \sum_{i=1}^{n} \max\{6\delta + 6\epsilon, a_{\sigma(i+1)} + 4\delta + 5\epsilon, b_{\sigma(i+1)} + 4\delta + 5\epsilon,$ $a_{\sigma(i+1)} + b_{\sigma(i+1)} + 2\delta + 4\epsilon\}$	$T^{\sigma}_{2,7} \geq T^{\sigma}_{2,14}$
$S_{2,8}$	$T^{\sigma}_{2,8} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{2,8} = \sum_{i=1}^{n} \max\{4\delta + 6\epsilon + \theta, a_{\sigma(i+1)} + 2\delta + 4\epsilon\} + \sum_{i=1}^{n} b_i$	$T^{\sigma}_{2,8} \geq T^{\sigma}_{2,5}$
$S_{2,9}$	$T^{\sigma}_{2,9} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{2,9} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon, a_{\sigma(i+1)} + 3\delta + 5\epsilon\} + \sum_{i=1}^{n} b_i$	$T^{\sigma}_{2,9} \geq T^{\sigma}_{2,5}$
$S_{2,10}$	$T^{\sigma}_{2,10} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{2,10} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon, b_{\sigma(i+1)} + 3\delta + 5\epsilon\} + \sum_{i=1}^{n} a_i$	$T^{\sigma}_{2,10} \geq T^{\sigma}_{2,3}$
$S_{2,11}$	$T^{\sigma}_{2,11} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{2,11} = \sum_{i=1}^{n}(4\delta + 6\epsilon + a_{\sigma(i+1)} + b_{\sigma(i+1)})$	$T^{\sigma}_{2,11} \geq T^{\sigma}_{2,3}$
$S_{2,12}$	$T^{\sigma}_{2,12} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{2,12} = \sum_{i=1}^{n} \max\{a_{\sigma(i+1)} + 2\epsilon + \theta, b_{\sigma(i)} + 4\delta + 6\epsilon + \theta\}$	$T^{\sigma}_{2,12} \geq T^{\sigma}_{1,14}$
$S_{2,13}$	$T^{\sigma}_{2,13} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{2,13} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon + \theta, a_{\sigma(i+1)} + 2\epsilon + \theta, b_{\sigma(i)} + 3\delta + 5\epsilon + \theta\}$	$T^{\sigma}_{2,13} \geq T^{\sigma}_{3,4}$
$S_{2,14}$	$T^{\sigma}_{2,14} = \sum_{i=1}^{n} T^{1\sigma(i)\sigma(i+1)}_{2,14} = \sum_{i=1}^{n} \max\{6\delta + 6\epsilon, a_{\sigma(i+1)} + 2\delta + 3\epsilon, b_{\sigma(i)} + 3\delta + 4\epsilon\}$	$T^{\sigma}_{2,14} \geq T^{\sigma}_{3,5}$
$S_{2,15}$	$T^{\sigma}_{2,15} = T^{\sigma}_{1,18}$	$T^{\sigma}_{2,15} \geq T^{\sigma}_{1,18}$
$S_{2,16}$	$T^{\sigma}_{2,16} = T^{\sigma}_{1,13}$	$T^{\sigma}_{2,16} \geq T^{\sigma}_{1,13}$

Table 7.2. Case 2: Cycle-Time Expressions for Scheduling Multiple-Part-Type Production Using CRM Sequences.

CRM	Cycle Time	Dominance
$S_{3,1}$	$T^{\sigma}_{3,1} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,1} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon, a_{\sigma(i+2)} + 3\delta + 5\epsilon, b_{\sigma(i+1)} + 3\delta + 5\epsilon\}$	$T^{\sigma}_{3,1} \geq T^{\sigma}_{1,1}$
$S_{3,2}$	$T^{\sigma}_{3,2} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,2} = \sum_{i=1}^{n} (3\delta + 6\epsilon + a_{\sigma(i+1)} + b_{\sigma(i+1)}) = T^{\sigma}_{1,13}$	$T^{\sigma}_{3,2} \geq T^{\sigma}_{1,13}$
$S_{3,3}$	$T^{\sigma}_{3,3} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,3} = \sum_{i=1}^{n} \max\{a_{\sigma(i+2)} + 4\delta + 6\epsilon, b_{\sigma(i+1)} + 2\delta + 4\epsilon\}$	UD
$S_{3,4}$	$T^{\sigma}_{3,4} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,4} = \sum_{i=1}^{n} \max\{4\delta + 6\epsilon + \theta, a_{\sigma(i+2)} + \theta + 2\epsilon - w_2^i, b_{\sigma(i+1)} + 2\delta + 4\epsilon\}$ $w_2^i = \max\{0, b_{\sigma(i)} - \max\{2\delta + 2\epsilon + \theta, a_{\sigma(i+1)} - w_2^{i-1} - 2\delta - 2\epsilon + \theta\}\}$	UD
$S_{3,5}$	$T^{\sigma}_{3,5} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,5} = \sum_{i=1}^{n} \max\{6\delta + 6\epsilon, a_{\sigma(i+2)} + 2\delta + 3\epsilon - w_2^i, b_{\sigma(i+1)} + 2\delta + 3\epsilon\}$ $w_2^i = \max\{0, b_{\sigma(i)} - \max\{4\delta + 3\epsilon, a_{\sigma(i+1)} - w_2^{i-1}\}\}$	UD
$S_{3,6}$	$T^{\sigma}_{3,6} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,6} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon + \theta, a_{\sigma(i+2)} + 2\epsilon + \theta - w_2^i, b_{\sigma(i+1)} + 2\delta + 3\epsilon\}$ $w_2^i = \max\{0, b_{\sigma(i)} - \max\{3\delta + 3\epsilon + \theta, a_{\sigma(i+1)} - w_2^{i-1} - 2\delta - \epsilon + \theta\}\}$	UD
$S_{3,7}$	$T^{\sigma}_{3,7} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,7} = \sum_{i=1}^{n} \max\{a_{\sigma(i+2)} + 5\delta + 6\epsilon, b_{\sigma(i+1)} + 2\delta + 3\epsilon\}$	$T^{\sigma}_{3,7} \geq T^{\sigma}_{1,1}$
$S_{3,8}$	$T^{\sigma}_{3,8} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,8} = \sum_{i=1}^{n} \max\{6\delta + 6\epsilon, a_{\sigma(i+1)} + 3\delta + 4\epsilon, b_{\sigma(i)} + 3\delta + 4\epsilon\} = T^{\sigma}_{1,18}$	$T^{\sigma}_{3,8} \geq T^{\sigma}_{1,18}$
$S_{3,9}$	$T^{\sigma}_{3,9} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,9} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon + \theta, a_{\sigma(i+2)} + 2\delta + 3\epsilon - w_2^i, b_{\sigma(i+1)} + \theta + 2\epsilon\}$ $w_2^i = \max\{0, b_{\sigma(i)} - \max\{5\delta + 4\epsilon, a_{\sigma(i+1)} - w_2^{i-1} + 2\delta + \epsilon - \theta\}\}$	UD
$S_{3,10}$	$T^{\sigma}_{3,10} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,10} = \sum_{i=1}^{n} \max\{3\delta + 6\epsilon + 2\theta, a_{\sigma(i+2)} + \theta + 2\epsilon - w_2^i, b_{\sigma(i+1)} + \theta + 2\epsilon\}$ $w_2^i = \max\{0, b_{\sigma(i)} - \max\{3\delta + 4\epsilon + \theta, a_{\sigma(i+1)} - w_2^{i-1}\}\}$	UD
$S_{3,11}$	$T^{\sigma}_{3,11} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,11} = \sum_{i=1}^{n} \max\{a_{\sigma(i+2)} + 3\delta + 6\epsilon + \theta, b_{\sigma(i+1)} + 2\epsilon + \theta\}$	UD
$S_{3,12}$	$T^{\sigma}_{3,12} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,12} = \sum_{i=1}^{n} \max\{3\delta + 6\epsilon + 3\theta, a_{\sigma(i+2)} + \theta + 2\epsilon - w_2^i, b_{\sigma(i+1)} + \theta + 2\epsilon\}$ $w_2^i = \max\{0, b_{\sigma(i)} - \max\{3\delta + 4\epsilon + 2\theta, a_{\sigma(i+1)} - w_2^{i-1}\}\}$	$T^{\sigma}_{3,12} \geq T^{\sigma}_{3,10}$
$S_{3,13}$	$T^{\sigma}_{3,13} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,13} = \sum_{i=1}^{n} \max\{a_{\sigma(i+2)} + 3\delta + 6\epsilon + 2\theta, b_{\sigma(i+1)} + 2\epsilon + \theta\}$	$T^{\sigma}_{3,13} \geq T^{\sigma}_{3,11}$
$S_{3,14}$	$T^{\sigma}_{3,14} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,14} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon + \theta, a_{\sigma(i+2)} + 3\delta + 5\epsilon + \theta, b_{\sigma(i+1)} + 2\epsilon + \theta\}$	$T^{\sigma}_{3,14} \geq T^{\sigma}_{3,9}$
$S_{3,15}$	$T^{\sigma}_{3,15} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,15} = \sum_{i=1}^{n} \max\{a_{\sigma(i+2)} + 4\delta + 6\epsilon + \theta, b_{\sigma(i+1)} + 2\epsilon + \theta\}$	$T^{\sigma}_{3,15} \geq T^{\sigma}_{3,11}$
$S_{3,16}$	$T^{\sigma}_{3,16} = \sum_{i=1}^{n} T^{2\sigma(i)\sigma(i+1)}_{3,16} = \sum_{i=1}^{n} \max\{4\delta + 6\epsilon + \theta, a_{\sigma(i+2)} + 2\delta + 4\epsilon - w_2^i, b_{\sigma(i+1)} + 2\epsilon + \theta\}$ $w_2^i = \max\{0, b_{\sigma(i)} - \max\{4\delta + 4\epsilon, a_{\sigma(i+1)} - w_2^{i-1} + 2\delta + 2\epsilon - \theta\}\}$	UD

Table 7.3. Case 3: Cycle-Time Expressions for Scheduling Multiple-Part-Type Production Using CRM Sequences.

7.2 Two-Machine Cells: Complexity

In this section we focus on the computational complexity of $RF_2^2|(free, A, MP, S_{u,v} \in \mathcal{S})|\mu$. We prove that the recognition version of the part scheduling problem subject to a given CRM sequence is strongly NP-complete when $S_{u,v} \in \widetilde{\mathcal{S}} = \{S_{3,4}, S_{3,5}, S_{3,6}, S_{3,9}, S_{3,10}, S_{3,16}\}$. For the remaining seven sequences in $\mathcal{S}$, an optimal part schedule can be found in polynomial time.

We first observe that the family $\widetilde{\mathcal{S}}$ contains those undominated CRM sequences for which the cycle time expressions (see Table 7.3) involve the recursive calculations of robot waiting times. We first consider the problem of calculating robot waiting times subject to a given part schedule σ and a robot move sequence $S_{u,v} \in \widetilde{\mathcal{S}}$, and derive a set of useful results for this problem via linear programming. We then show that for a given part schedule σ, the cycle time $T_{u,v}^{\sigma}$ for an MPS cycle processing under the CRM sequence $S_{u,v} \in \widetilde{\mathcal{S}}$ can be calculated in polynomial time. This is not obvious because of the recursive expressions for w_2^i in the corresponding cycle time expressions (Table 7.3). We then prove that the recognition version of the part scheduling problem under any CRM sequence $S_{u,v} \in \widetilde{\mathcal{S}}$ is strongly NP-complete.

7.2.1 Cycle Time Calculation

In the proof of Lemma 7.1, we derived expressions to calculate the robot waiting times for MPS parts processed under the CRM sequence $S_{3,10}$. Following the same procedure, one may easily obtain the cycle time expressions for the remaining sequences in the set $\widetilde{\mathcal{S}}$.

Given a sequence $S_{u,v} \in \widetilde{\mathcal{S}}$, define the values $Z_{u,v}$, $\gamma_{u,v}^a$, and $\gamma_{u,v}^b$ as in Table 7.4. It is then easy to verify that for $RF_2^2|(free, A, MP, S_{u,v} \in \widetilde{\mathcal{S}})|\mu$, the calculation of the cycle time $T_{u,v}^{\sigma}$ of MPS production in a given part schedule σ admits the following formulation:

$$T_{u,v}^{\sigma} = Z_{u,v} + \sum_{i=1}^{n} (w_1^i + w_2^i), \tag{7.4}$$

$$w_1^i = \max\{0, a_{\sigma(i)} - \gamma_{u,v}^a - w_2^{i-2}\}, \quad i = 1, \ldots, n, \tag{7.5}$$

$$w_2^i = \max\{0, b_{\sigma(i)} - \gamma_{u,v}^b - w_1^{i+1}\}, \quad i = 1, \ldots, n, \tag{7.6}$$

$$w_1^i = w_1^{i \pm n}, \quad w_2^i = w_2^{i \pm n}, \quad i = 1, \ldots, n. \tag{7.7}$$

We then note that equations (7.5) and (7.6) can be combined into a set of $2n$ equations of the following form:

$$x_i = \max\{0, \beta_i - x_{i+1}\}, \qquad i = 1, \ldots, 2n,$$

where

$$\left\{\begin{array}{rl} x_{2i-1} & = w_1^{n+1-i}, \quad x_{2i} = w_2^{n-1-i}, \quad i = 1, \ldots, n, \\ \beta_{2i-1} & = a_{\sigma(n+1-i)} - \gamma_{u,v}^a, \quad i = 1, \ldots, n, \\ \beta_{2i} & = b_{\sigma(n-1-i)} - \gamma_{u,v}^b, \quad i = 1, \ldots, n, \\ x_i & = x_{i\pm 2n}; \quad \beta_i = \beta_{i\pm 2n}, \quad i = 1, \ldots, 2n, \\ a_{\sigma(i)} & = a_{\sigma(i\pm n)}, \quad b_{\sigma(i)} = b_{\sigma(i\pm n)}, \quad i = 1, \ldots, n, \\ w_1^i & = w_1^{i\pm n},\ w_2^i = w_2^{i\pm n},\ i = 1, \ldots, n. \end{array}\right. \tag{7.8}$$

Consequently, expression (7.4) translates into $T_{u,v}^{\sigma} = Z_{u,v} + \sum_{i=1}^{2n} x_i$. Let $r = 2n$. The problem (7.5)-(7.7) of finding robot waiting times can then be reformulated as follows:

Problem P1: Given a vector $\beta \in R^r$, find a vector $x \in R^r$ satisfying the following conditions:

$$x_i = \max\{0, \beta_i - x_{i+1}\}, \quad i = 1, \ldots, r, \tag{7.9}$$

$$x_{r+i} = x_i, \quad \beta_{r+i} = \beta_i.$$

Because the x_i's in equations (7.9) are nested, the issue of the existence of a solution to Problem P1 must be examined. Consider the following problem:

Problem P2 : Given a vector $\beta \in R^r$, find vectors $x, s \in R^r$ such that

CRM $S_{u,v}$	$Z_{u,v}$	$\gamma_{u,v}^a$	$\gamma_{u,v}^b$
$S_{3,4}$	$6n\epsilon + 4n\delta + n\theta$	$4\epsilon + 4\delta$	$2\epsilon + 2\delta + \theta$
$S_{3,5}$	$6n\epsilon + 6n\delta$	$3\epsilon + 4\delta$	$3\epsilon + 4\delta$
$S_{3,6}$	$6n\epsilon + 5n\delta + n\theta$	$4\epsilon + 5\delta$	$3\epsilon + 3\delta + \theta$
$S_{3,9}$	$6n\epsilon + 5n\delta + n\theta$	$3\epsilon + 3\delta + \theta$	$4\epsilon + 5\delta$
$S_{3,10}$	$6n\epsilon + 3n\delta + 2n\theta$	$4\epsilon + 3\delta + \theta$	$4\epsilon + 3\delta + \theta$
$S_{3,16}$	$6n\epsilon + 4n\delta + n\theta$	$2\epsilon + 2\delta + \theta$	$4\epsilon + 4\delta$

Table 7.4. The Values of $Z_{u,v}$, $\gamma_{u,v}^a$, and $\gamma_{u,v}^b$ for a Given CRM Sequence $S_{u,v} \in \widetilde{\mathcal{S}}$.

$$x_i + x_{i+1} - s_i = \beta_i, \quad i = 1, \ldots, r, \tag{7.10}$$

$$x_i \geq 0, \quad s_i \geq 0, \qquad i = 1, \ldots, r, \tag{7.11}$$

$$x_i s_i = 0, \qquad i = 1, \ldots, r, \tag{7.12}$$

$$x_{r+i} = x_i, \quad s_{r+i} = s_i, \quad \beta_{r+i} = \beta_i.$$

In Lemma 7.2, we establish the connection between Problems P1 and P2.

LEMMA 7.2 *Let $\beta \in R^r$ be any given vector. The vector $x^0 \in R^r$ is a solution to Problem P1 iff the pair of vectors $x^0, s^0 \in R^r$ is a solution to Problem P2.*

Proof. *If part:* Let x_i^0, s_i^0, $i = 1, \ldots, r$, be a solution to Problem P2. We aim to show that x_i^0, $i = 1, \ldots, r$, is a solution to Problem P1. We have two possible cases: $x_i^0 > 0$ and $x_i^0 = 0$. Suppose that for some i, we have $x_i^0 > 0$. Then, by (7.12) we have $s_i^0 = 0$, which by (7.10) implies $\beta_i - x_{i+1}^0 = x_i^0$. Since $x_i^0 > 0$, we obtain $\max\{0, \beta_i - x_{i+1}^0\} = \beta_i - x_{i+1}^0 = x_i^0$, and thus condition (7.9) holds. Consider now the case when $x_i^0 = 0$, for some i. Then from (7.10)–(7.11), we have $s_i^0 = x_{i+1}^0 - \beta_i \geq 0$. Therefore, $\beta_i - x_{i+1}^0 \leq 0$, and so $\max\{0, \beta_i - x_{i+1}^0\} = 0$. Thus, we have $x_i^0 = \max\{0, \beta_i - x_{i+1}^0\}$, and condition (7.9) holds.
Only if part : Assume that x_i^0, $i = 1, \ldots, r$, is a solution to Problem P1. We define $s_i^o = x_i^o + x_{i+1}^o - \beta_i$, $i = 1, \ldots, r$, and prove that x_i^0, s_i^0, $i = 1, \ldots, r$, solve Problem P2. By (7.9), $x_i^o \geq 0$ and $x_i^o + x_{i+1}^o \geq \beta_i$, for all i. The second inequality yields $s_i^o \geq 0$, and thus the conditions (7.10)–(7.11) are met. It then remains to be shown that x_i^0, s_i^0, $i = 1, \ldots, r$, satisfy (7.12). If for some i, $s_i^o = 0$, then (7.12) obviously holds for the pair x_i^0, s_i^0. Suppose that $s_i^o > 0$ for some i. We then have $x_i^o + x_{i+1}^o - \beta_i > 0$. This implies $x_i^o > \beta_i - x_{i+1}^o$, which in turn yields $x_i^o = 0$, and so (7.12) holds. ∎

Lemma 7.2 can be used now to show that Problem P1 has a solution.

LEMMA 7.3 *There always exists a solution to Problem P1.*

Proof. Problem P2 can be rewritten as the following Linear Complementarity Problem (LCP): Given a vector $\beta \in R^r$ and a matrix $A \in R^{r \times r}$, find vectors $x, s \in R^r$ such that

$$Ax - Is = \beta, \tag{7.13}$$

$$x \geq 0, \quad s \geq 0, \tag{7.14}$$

$$x's = 0, \tag{7.15}$$

where x' is the transpose of x and

$$A = \begin{pmatrix} 1 & 1 & 0 & & 0 & 0 \\ 0 & 1 & 1 & \cdot\cdot & 0 & 0 \\ 0 & 0 & 1 & \cdot\cdot & 0 & 0 \\ & \cdot\cdot & \cdot\cdot & \cdot\cdot & \cdot\cdot & \\ 0 & 0 & 0 & \cdot\cdot & 1 & 1 \\ 1 & 0 & 0 & & 0 & 1 \end{pmatrix}. \tag{7.16}$$

This matrix $A = (a_{i,j})_{r\times r}$ satisfies the following property:

$$a_{i,j} \geq 0 \text{ for all } i,j = 1,\ldots,r, \text{ and } a_{i,i} > 0 \text{ for all } i = 1,\ldots,r, \tag{7.17}$$

which characterizes a class of matrices for which LCPs are known to have solutions (Cottle et al. [37]). Thus, by Lemma 7.2, Problem P1 has a solution. ∎

The following lemma is used in the development of a polynomial-time algorithm to find a solution to Problem P1.

LEMMA 7.4 *When r is even, Problem P1 has a solution with at least one $x_i = 0$.*

Proof. When r is even, the matrix A given by (7.16) is a singular matrix with rank $(r-1)$. Thus, there exists a solution (x^o,s^o) for LCP Problem P2 with at least one $x_i^o = 0$. The result now follows from Lemma 7.2. ∎

For $RF_2^2|(free,A,\ MP,S_{u,v}\in\widetilde{\mathcal{S}})|\mu$, we now present a polynomial-time algorithm that finds the cycle time $\left(T_{u,v}^{\sigma}\right)$ and the corresponding robot waiting times $(w_1^i,\ w_2^i,\ i = 1,\ldots,n)$ for each $S_{u,v}\in\widetilde{\mathcal{S}}$ and a given part schedule σ. The idea behind the algorithm is as follows. We have proven in Lemma 7.4 that a solution exists with at least one zero robot waiting time. A steady-state solution can then be found by setting robot wait times (each of $2n$, in the worst case) to be equal to zero and checking whether it produces a steady state cyclic solution satisfying condition (7.9).

Algorithm RobotWait

Input: A CRM sequence $S_{u,v} \in \widetilde{\mathcal{S}}$ and a part schedule $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ of parts $P_k \in \mathcal{P}$.

Output: Waiting times w_1^i and w_2^i, $i = 1, \ldots, n$, and the cycle time $T_{u,v}^{\sigma}$.

Step 1: Find values of $Z_{u,v}$, $\gamma_{u,v}^a$, and $\gamma_{u,v}^b$ defined in Table 7.4.

For $i = 1, \ldots, n$: set $\beta_{2i-1} = a_{\sigma(n+1-i)} - \gamma_{u,v}^a$ and $\beta_{2i} = b_{\sigma(n-1-i)} - \gamma_{u,v}^b$. Set $j = 1$.

Step 2: Set $x_j = 0$.

For $i = j-1, \ldots, 1$: find $x_i = \max\{0, \beta_i - x_{i+1}\}$.

If $j \neq 2n$, then find $x_{2n} = \max\{0, \beta_{2n} - x_1\}$.

For $i = 2n-1, \ldots, j+1$: find $x_i = \max\{0, \beta_i - x_{i+1}\}$.

Step 3: If $j = 2n$, then set $x^* = \max\{0, \beta_{2n} - x_1\}$; otherwise set $x^* = \max\{0, \beta_j - x_{j+1}\}$

Step 4: If $x^* = 0$, then calculate $T_{u,v}^{\sigma} = Z_{u,v} + \sum_{i=1}^{2n} x_i$ and find $w_1^{n+1-i} = x_{2i-1}$ and $w_2^{n-1-i} = x_{2i}$ for $i = 1, \ldots, n$, and Stop. Otherwise set $j = j + 1$, and go to Step 2.

The running time of algorithm RobotWait is $O(n^2)$.

We next formulate a linear program to calculate cycle time $T_{u,v}^{\sigma}$ and show how it is related to Problem P1. This LP formulation is also used in Section 7.3 to prove the dominance of a particular robot move sequence for practically relevant special cases.

Problem P3:

$$\min \sum_{i=1}^{r} x_i \tag{7.18}$$

$$\text{s.t.} \quad x_i + x_{i+1} \geq \beta_i, \quad i = 1, \ldots, r, \tag{7.19}$$

$$x_i \geq 0, \quad i = 1, \ldots, r, \tag{7.20}$$

$$x_{r+1} = x_1, \quad \beta_{r+1} = \beta_1.$$

Note that the solution to P3 is also a practically implementable steady-state solution for our robotic system. For the cycle time computation,

Crama et al. [39] provide an $O(n^2)$ algorithm based on the cyclic PERT approach as an alternative to the above LP approach.

LEMMA 7.5 *Every solution to Problem P1 is a solution to Problem P3.*

Proof. Let $x^0 \in R^r$ be a solution to Problem P1. It is trivial to verify that x^0 satisfies constraints (7.19)–(7.20), so we are only left to prove that x^0 achieves the minimum of the objective function (7.18).

We start with the obvious lower bound on the value of $\sum_{i=1}^{r} x_i$ in (7.18). Let x_i^*, $i = 1, \ldots, r$, be a solution to Problem P3. Constraints (7.19)–(7.20) imply that

$$\sum_{i=1}^{r} x_i^* \geq \sum_{i=1}^{r} \frac{\beta_i}{2}. \tag{7.21}$$

By Lemma 7.2, the pair of vectors x^0 and s^0, with s^0 defined by (7.10), gives a solution to Problem P2. Clearly, if all $s_i^o = 0$, $i = 1, \ldots, r$, we have that $2\sum_{i=1}^{r} x_i^0 = \sum_{i=1}^{r} \beta_i$ (by (7.10)), and the lemma's claim follows immediately due to (7.21). Hence, we only need to consider the case when there is at least one s_i^0 of strictly positive value. Thus, from now on we assume that there exists at least one s_i^0, $i \in \{1, \ldots, r\}$, such that $s_i^0 > 0$. Consider the dual of Problem P3.

Problem P4:

$$\max \sum_{i=1}^{r} \beta_i y_i \tag{7.22}$$

$$\text{s.t.} \qquad y_i + y_{i-1} \leq 1, \quad i = 1, \ldots, r, \tag{7.23}$$

$$y_i \geq 0, \quad i = 1, \ldots, r, \tag{7.24}$$

$$y_0 = y_r.$$

To prove the claim of the lemma, it is sufficient to find a vector $y^o \in R^r$ satisfying conditions (7.23)–(7.24) for which the following relation holds:

$$\sum_{i=1}^{r} \beta_i y_i^0 = \sum_{i=1}^{r} x_i^0. \tag{7.25}$$

Let $I \subseteq \{1, 2, \ldots, r\}$ be the set of all indices i such that $s_i^0 > 0$ (i.e., $s_i^0 > 0$ for $i \in I$, and $s_i^0 = 0$ for $i \notin I$). Let $I = \{i_1, i_2, \ldots, i_\ell\}$ and

$i_1 < i_2 < \ldots < i_\ell$. For the sake of notational simplicity, we will give the proof for the case when $i_\ell = r$. Due to the cyclic nature of problem, the latter assumption can always be realized by setting $\overline{x}^0_{i+r-i_\ell} = x^0_i$, $i = 1, \ldots, i_\ell$, and $\overline{x}^0_{i-i_\ell} = x^0_i$, $i = i_\ell + 1, \ldots, r$, and working with new variables $\overline{x}^0_i$ instead of x^0_i.

Given a set I, we define $i_0 = 0$ and find the values k_j, $j = 1, \ldots, \ell$, as follows

$$k_j = \left\lceil \frac{i_j - i_{j-1} - 1}{2} \right\rceil .$$

Furthermore, we define the sets of indexes $K_j = \{i_{j-1} + 1, i_{j-1} + 3, \ldots, i_{j-1} + 2k_j - 1\}$, $j = 1, \ldots, \ell$, and $K = K_1 \cup K_2 \cup \ldots \cup K_\ell$. We are now ready to introduce a vector y^0 which is a solution to Problem P4.

We define the vector $y^0 \in R^r$ as follows. For $j \in \{1, \ldots, r\}$ we set $y^0_i = 1$ if $i \in K$, and $y^0_i = 0$ otherwise. The condition (7.24) is then satisfied. Furthermore, by construction of sets K_j, $j = 1, \ldots, \ell$, the set K does not contain any two consecutive indices i and $i + 1$, and, hence, the condition (7.23) is met. For each $i \in I$, we have $s^0_i > 0$, so $x^0_i = 0$. Additionally, for each $y^0_i = 1$ we have $s^0_i = 0$, which yields $x^0_i + x^0_{i+1} = \beta_i$ for each $i \in K$ (by (7.10)). Moreover, by construction of set K, all non-zero x^0_i are included into the set of equalities $\left\{x^0_i + x^0_{i+1} = \beta_i\right\}_{i \in K}$. With no two consecutive indices i and $i+1$ being included into set K, we thus obtain

$$\sum_{i=1}^{r} \beta_i y^0_i = \sum_{i \in K} \beta_i y^0_i = \sum_{i \in K} \left(x^0_i + x^0_{i+1}\right) = \sum_{i=1}^{r} x^0_i.$$

Hence the relation (7.25) is established. This completes the proof. ∎

REMARK 7.1 The converse of Lemma 7.5 is not always true. For example, if $r = 4$, $\beta_1 = 4$, $\beta_2 = 6$, $\beta_3 = 3$, $\beta_4 = 2$, a solution to Problem P3 is $x_1 = 0$, $x_2 = 4$, $x_3 = 2$, and $x_4 = 2$, which is not a solution to Problem P1.

7.2.2 Strong NP-Completeness Results

Algorithm RobotWait always finds a solution for steady-state robot waiting times in polynomial time. This implies the following result.

THEOREM 7.1 *The recognition version of the* $RF^2_2|(free,A,MP,S_{u,v} \in \widetilde{\mathcal{S}})|\mu$ *part scheduling problem is in the class NP.*

The theorem below establishes that the problem of finding an optimal part schedule in a two-machine dual-gripper robotic cell is intractable.

THEOREM 7.2 *The recognition version of the* $RF_2^2|(free,A,MP,S_{3,10})|\mu$ *part scheduling problem is strongly NP-complete.*

Proof. The proof is similar to the NP-completeness results shown in Hall et al. [76] and in Chapter 6. The reduction is from Numerical Matching with Target Sums (NMTS) (see Chapter 6).

Given an arbitrary instance of NMTS, consider the following instance of the problem: The MPS consists of three types of parts, where each type consists of s parts: $\mathcal{P} = \{Px_i, Py_i, Pz_i | 1 \leq i \leq s\}$ and $n = 3s$. The processing times for part Px_i on M_1 and M_2 are K and $2K + x_i$, respectively; for part Py_i they are K and $2K - y_i$, respectively, and for part Pz_i they are $3K + z_i$ and K, respectively, for $i = 1, \ldots, s$, where $K = X$. It is also assumed that $X = Y + Z$, where $X = \sum x_i, Y = \sum y_i$, and $Z = \sum z_i$. The system parameters are $\delta = K/3$, $\epsilon = 0$, and $\theta = 0$. The decision problem can be stated as follows: "Does there exist a schedule with cycle time $C_t \leq D = 5sK + X - Y$?".

Let $\alpha = 3\delta + 4\epsilon + \theta$. Note that $\alpha = K$ and $K > y_i$, $K > x_i$, $i = 1, \ldots, s$.

($\Rightarrow$) Suppose that a solution to NMTS exists. Consider the following MPS schedule σ=[Py_1, Px_1, Pz_1, Py_2, Px_2, Pz_2, ..., Py_s, Px_s, Pz_s]. The Gantt chart for this schedule can be viewed as concurrently processing parts on three machines, as shown in Figure 7.1, where the processing times of the parts on the three machines, denoted M_0', M_1', and M_2', are given. Note that M_0', M_1', and M_2' are equivalent to the robot, M_1, and M_2, respectively, in the robotic cell.

Also note that the processing times on M_1' are a function of w_2^i. From Lemma 7.1, we have $w_2^1 = \max\{0, 2K - y_1 - \max\{K, K - w_2^{3s}\}\} = \max\{0, K - y_1\} = K - y_1$, $w_2^2 = 0, w_2^3 = 0, w_2^4 = K - y_2, w_2^5 = 0, w_2^6 = 0, \ldots, w_2^{3s-2} = K - y_s, w_2^{3s-1} = 0$, and $w_2^{3s} = 0$. By substituting these values of waiting times into the processing times, we get the processing times on M_2' (Figure 7.2). Hence, the cycle time $C_t = s(2K) - \sum_{i=1}^{s} y_i + s(2K) + \sum_{i=1}^{s} x_i + s(K) = 5sK + X - Y$.

Figure 7.1. Sequence $\ldots, Py_1, Px_1, Pz_1, Py_2, Px_2, Pz_2, \ldots$

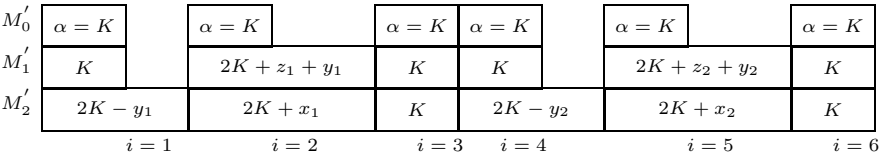


Figure 7.2. Sequence $\ldots, Py_1, Px_1, Pz_1, Py_2, Px_2, Pz_2, \ldots$

($\Leftarrow$) Suppose there exists an MPS schedule σ_0 such that $C_t \leq D$. We now show that σ_0 is concatenation of s subschedules as follows: $\sigma_0 = \{\rho_1, \rho_2, \ldots, \rho_s\}$, where $\rho_k = [Py_t, Px_u, Pz_v]$ or $\rho_k = [Px_u, Py_t, Pz_v]$, $k = 1, \ldots, s$. Furthermore, if $C_t \leq D$, then there must exist a solution to NMTS. To do so, we present and prove a series of facts about schedule σ_0.

Fact 1. Machine M_2 is busy processing parts throughout schedule σ_0.

The total processing time of all the parts on M_2 in an MPS cycle is $5sK + X - Y$. Since $D = 5sK + X - Y$, any idle time on M_2 implies that $C_t > D$.

Fact 2. In σ_0, $\max_{1 \leq i \leq 3s}\{w_2^i\} \leq K + \max_{1 \leq i \leq s}\{x_i\}$.

From Lemma 7.1, $w_2^i = \max\{0, b_{\sigma(i)} - \max\{3\delta + 4\epsilon + \theta, a_{\sigma(i+1)} - w_2^{i-1}\}\}$. Therefore, $\max_{1 \leq i \leq 3s}\{w_2^i\} \leq \max_{1 \leq i \leq 3s}\{b_{\sigma(i)}\} - 3\delta - 4\epsilon - \theta = 2K + \max_{1 \leq i \leq s}\{x_i\} - 3\delta - 4\epsilon - \theta = K + \max_{1 \leq i \leq s}\{x_i\}$.

Fact 3. σ_0 is concatenation of s subschedules as follows: $\sigma_0 = \{\rho_1, \rho_2, \ldots, \rho_s\}$, where $\rho_k = [Py_t, Px_u, Pz_v]$ or $\rho_k = [Px_u, Py_t, Pz_v]$, $k = 1, \ldots, s$.

Without loss of generality, we may assume that the third part scheduled is Pz_1. We need to show that the previous two parts scheduled immediately before Pz_1 are Px_i, Py_i or Py_i, Px_i. To prove Fact 3, we establish that the following 5 cases of subschedules are not possible in schedule σ_0.

Case 1. $[Pz_i, Pz_1]$: An idle time of at least $K + z_1 - \max_{1 \leq i \leq s}\{x_i\}$ occurs on M_2' (see Figure 7.3), contradicting Fact 1.

Case 2. $[Py_i, Py_k, Pz_1]$: An idle time of $z_1 + y_i + y_k$ occurs on M_2' (see Figure 7.4), contradicting Fact 1.

Case 3. $[Pz_i, Px_i, Pz_1]$: An idle time of $K + z_1 - x_i$ occurs on M_2' (see Figure 7.5), contradicting Fact 1.

$\bar{h}$=idle time on M_2'=$3K + z_1 - w_2^1 - K$
$\geq 2K + z_1 - (K + \max_{1\leq i\leq s}\{x_i\})$ since $w_2^1 \leq K + \max_{1\leq i\leq s}\{x_i\}$ from Fact 2
$\geq K + z_1 - \max_{1\leq i\leq s}\{x_i\} > 0.$

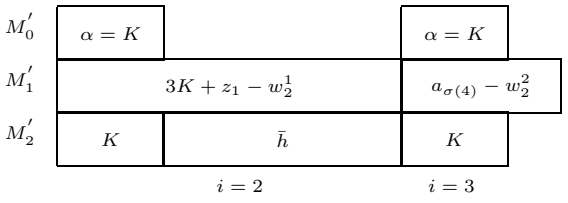


Figure 7.3. Subschedule $\ldots, Pz_i, Pz_1, \ldots$

$w_2^1 = \max\{0, 2K - y_i - \max\{K, K - w_2^{3s}\}\} = K - y_i$
$\Rightarrow \bar{h} = 3K + z_1 - w_2^1 - 2K + y_k$
$= K + z_1 + y_k - w_2^1$
$= z_1 + y_i + y_k > 0.$

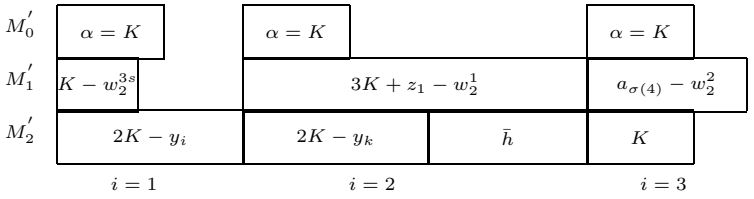


Figure 7.4. Subschedule $\ldots, Py_i, Py_k, Pz_1, \ldots$

$w_2^1 = \max\{0, K - \max\{K, K - w_2^{3s}\}\} = 0$
$\Rightarrow \bar{h} = 3K + z_1 - w_2^1 - (2K + x_i)$
$= K + z_1 - x_i - w_2^1$
$= K + z_1 - x_i > 0.$

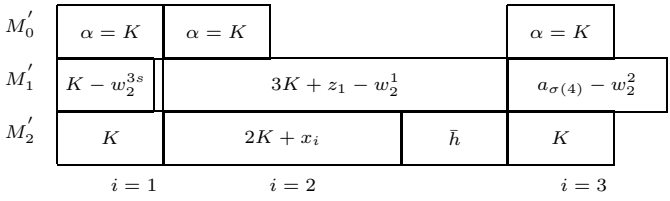


Figure 7.5. Subschedule $\ldots, Pz_i, Px_i, Pz_1, \ldots$

Case 4. $[Pz_i, Py_k, Pz_1]$: An idle time of $K + z_1 + y_k$ occurs on M_2' (see Figure 7.6), contradicting Fact 1. Therefore, the remaining possible subschedules are $[Px_i, Px_k, Pz_1]$, $[Py_i, Px_i, Pz_1]$, and $[Px_i, Py_i, Pz_1]$.
Case 5. $[Px_i, Px_k, Pz_1]$: There exists at least one $[Py_i, Py_k, Pz_1]$ subschedule, contradicting Case 2 above.

$w_2^1 = \max\{0, K - \max\{K, K - w_2^{3s}\}\} = 0$
$\Rightarrow \bar{h} = 3K + z_1 - w_2^1 - (2K - y_k)$
$= K + z_1 + y_k - w_2^1$
$= K + z_1 + y_k > 0.$

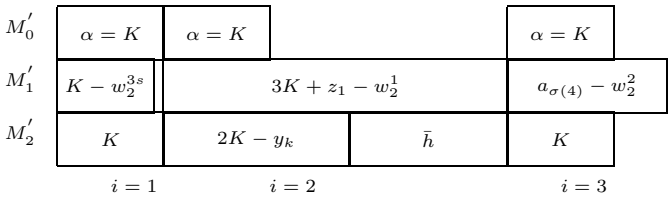


Figure 7.6. Subschedule $\ldots, Pz_i, Py_k, Pz_1, \ldots$

Therefore, σ_0 must take the following form: $\sigma_0 = \{\rho_1, \rho_2, \ldots, \rho_s\}$, where $\rho_k = [Py_t, Px_u, Pz_v]$ or $\rho_k = [Px_u, Py_t, Pz_v]$, $k = 1, \ldots, s$. This proves Fact 3. Furthermore, in these two subschedules (see Figures 7.7 and 7.8), if $x_i < y_i + z_1$, then an idle time of $z_1 + y_i - x_i$ occurs on M_2', contradicting Fact 1. If $x_i > y_i + z_1$, then there exists an index r such that $x_r < y_{h_r} + z_{i_r}$. Hence an idle time of $z_{i_r} + y_{h_r} - x_r > 0$ occurs on M_2', contradicting Fact 1. Therefore, $x_r = y_{h_r} + z_{i_r}$ for $r = 1, \ldots, s$. Thus, there exists a solution to NMTS. ∎

$w_2^1 = \max\{0, 2K + x_i - \max\{K, K - w_2^{3s}\}\} = K + x_i$
$\Rightarrow \bar{h} = K + z_1 - w_2^1 + y_i$
$= z_1 + y_i - x_i > 0.$

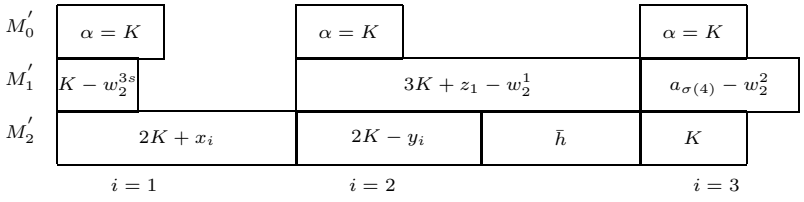


Figure 7.7. Subschedule $\ldots, Px_i, Py_i, Pz_1, \ldots$

THEOREM 7.3 *For each* $S_{u,v} \in \{S_{3,4}, S_{3,5}, S_{3,6}, S_{3,9}, S_{3,16}\}$, *the recognition version of the part scheduling problem* $RF_2^2|(free,A,MP,S_{u,v})|\mu$ *is strongly NP-complete.*

Proof. By Theorem 7.2, the part scheduling problem associated with the CRM sequence $S_{3,10}$ is NP-complete. This problem is reducible to a part scheduling problem associated with any $S_{u,v} \in \{S_{3,4}, S_{3,5}, S_{3,6}, S_{3,9},$

$S_{3,16}\}$, since the cycle time expression for each such problem takes the following form: $T^{\sigma}_{u,v} = \sum_{i=1}^{n} \max\{t_1, a_{\sigma(i+2)} + t_2 - w_2^i, b_{\sigma(i+1)} + t_3\}$, $w_2^i = \max\{0, b_{\sigma(i)} - \max\{z_1, a_{\sigma(i+1)} + z_2 - w_2^{i-1}\}\}$, where t_1, t_2, t_3, z_1, and z_2 are constants (see Table 7.3). ■

$w_2^1 = \max\{0, 2K - y_i - \max\{K, K - w_2^{3s}\}\} = K - y_i$
$\Rightarrow \bar{h} = 3K + z_1 - w_2^1 - 2K - x_i$
$= K + z_1 - x_i - w_2^1$
$= z_1 + y_i - x_i > 0.$

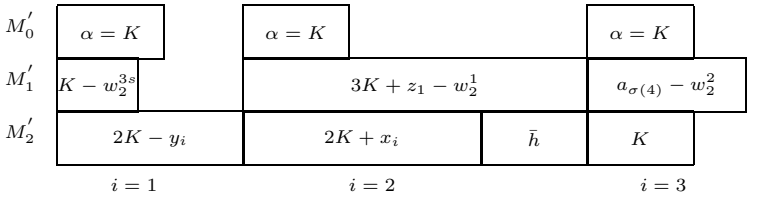


Figure 7.8. Subschedule $\ldots, Py_i, Px_i, Pz_1, \ldots$

7.2.3 Polynomially Solvable Problems

In the previous section, we showed that $RF_2^2|(free,A,MP,S_{u,v})|\mu$ is intractable for 6 out of the 13 undominated CRM sequences in $\mathcal{S}$. Here, we demonstrate that the part scheduling problems $RF_2^2|(free,\ A,MP,S_{u,v})|\mu$ associated with the remaining seven undominated CRM sequences admit an efficient solution.

THEOREM 7.4 *For $RF_2^2|(free,A,MP,S_{1,13})|\mu$, every part schedule σ is optimal.*

Proof. From Table 7.1, the cycle time for the CRM sequence $S_{1,13}$ is equal to $\sum_{i=1}^{n}(3\delta + 6\epsilon + a_{\sigma(i)} + b_{\sigma(i)})$, which is independent of σ. Thus, every part schedule σ is optimal. ■

THEOREM 7.5 *For $S_{u,v} \in \{S_{1,1}, S_{1,4}, S_{1,14}, S_{1,15}, S_{3,3}, S_{3,11}\}$, $RF_2^2|(free,\ A,MP,S_{u,v})|\mu$ can be solved in time $O(n \log n)$.*

Proof. From Tables 7.1 and 7.3, the cycle time expression for each $S_{u,v} \in \{S_{1,1}, S_{1,4}, S_{1,14}, S_{1,15}, S_{3,3}, S_{3,11}\}$ has the following common format: $\sum_{i=1}^{n} \max\{t_1, a_{\sigma(i+1)} + t_2, b_{\sigma(i)} + t_3\}$, where t_1, t_2, and t_3 are constants. Thus, each of these problems can be reduced to a problem of the form

$$\min_{\sigma} \left(\sum_{i=1}^{n} \max\{e_{\sigma(i+1)}, f_{\sigma(i)}\} \right),$$

which can be solved optimally in time $O(n \log n)$ by the Gilmore-Gomory algorithm (see Appendix B). ∎

Table 7.5 summarizes the results of this section.

7.3 Analyzing Two-Machine Cells with Small Gripper Switch Times

In this section, we consider a special case of a two-machine robotic cell with a dual-gripper robot in which the value of the gripper switch time θ is small relative to other problem parameters. For most robotic cells in practice, θ is quite small in comparison to processing times and robot move time; that is,

$$\theta \leq \min \left\{ \delta, \min_{i \in N} a_i, \min_{i \in N} b_i \right\}. \tag{7.26}$$

In this case, we identify a CRM sequence that gives the smallest cycle time for $RF_2^2|(free,A,MP,S_{u,v} \in \mathcal{S}_0)|\mu$.

THEOREM 7.6 *For $RF_2^2|(free,A,MP,S_{u,v} \in \mathcal{S}_0)|\mu$ under condition (7.26), the CRM sequence $S_{3,10}$ delivers an optimal solution, i.e., the bound*

$$T^*_{3,10} \leq T^*_{u,v} \tag{7.27}$$

holds true for any CRM sequence $S_{u,v}$.

Proof. Recall that $\mathcal{S} = \{S_{1,1}, S_{1,4}, S_{1,13}, S_{1,14}, S_{1,15}, S_{3,3}, S_{3,11}, S_{3,4}, S_{3,5}, S_{3,6}, S_{3,9}, S_{3,10}, S_{3,16}\}$ is the set of all 13 undominated CRM sequences for the part scheduling problem $RF_2^2|(free,A,MP,S_{u,v} \in \mathcal{S}_0)|\mu$. We prove the theorem by showing that $T^*_{3,10} \leq T^*_{u,v}$ for any CRM sequence $S_{u,v} \in \mathcal{S}$. Namely, we will demonstrate that for any sequence $S_{u,v} \in \mathcal{S} \setminus \{S_{3,10}\}$, the following is true: if $\sigma^*_{u,v}$ is a permutation that delivers the optimal value of $T^*_{u,v}$ for $S_{u,v} \in \mathcal{S} \setminus \{S_{3,10}\}$, then

$$T^*_{u,v} = T_{u,v}^{\sigma^*_{u,v}} \geq T_{3,10}^{\sigma^*_{u,v}} \geq T^*_{3,10}. \tag{7.28}$$

CRM	Cycle Time	Complexity
$S_{1,1}$	$T^{\sigma}_{1,1} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon, a_{\sigma(i+1)} + 3\delta + 5\epsilon, b_{\sigma(i)} + 2\delta + 3\epsilon\}$	P
$S_{1,4}$	$T^{\sigma}_{1,4} = \sum_{i=1}^{n} \max\{a_{\sigma(i+1)} + 2\delta + 4\epsilon, b_{\sigma(i)} + 4\delta + 6\epsilon\}$	P
$S_{1,13}$	$T^{\sigma}_{1,13} = \sum_{i=1}^{n} (3\delta + 6\epsilon + a_{\sigma(i)} + b_{\sigma(i)})$	P
$S_{1,14}$	$T^{\sigma}_{1,14} = \sum_{i=1}^{n} \max\{b_{\sigma(i)} + 3\delta + 6\epsilon + \theta, a_{\sigma(i+1)} + 2\epsilon + \theta\}$	P
$S_{1,15}$	$T^{\sigma}_{1,15} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon, a_{\sigma(i+1)} + 2\delta + 3\epsilon, b_{\sigma(i)} + 3\delta + 5\epsilon\}$	P
$S_{3,3}$	$T^{\sigma}_{3,3} = \sum_{i=1}^{n} \max\{a_{\sigma(i+1)} + 4\delta + 6\epsilon, b_{\sigma(i)} + 2\delta + 4\epsilon\}$	P
$S_{3,4}$	$T^{\sigma}_{3,4} = \sum_{i=1}^{n} \max\{4\delta + 6\epsilon + \theta, a_{\sigma(i+2)} + \theta + 2\epsilon - w_2^i, b_{\sigma(i+1)} + 2\delta + 4\epsilon\}$ $w_2^i = \max\{0, b_{\sigma(i)} - \max\{2\delta + 2\epsilon + \theta, a_{\sigma(i+1)} - w_2^{i-1} - 2\delta - 2\epsilon + \theta\}\}$	NPC
$S_{3,5}$	$T^{\sigma}_{3,5} = \sum_{i=1}^{n} \max\{6\delta + 6\epsilon, a_{\sigma(i+2)} + 2\delta + 3\epsilon - w_2^i, b_{\sigma(i+1)} + 2\delta + 3\epsilon\}$ $w_2^i = \max\{0, b_{\sigma(i)} - \max\{4\delta + 3\epsilon, a_{\sigma(i+1)} - w_2^{i-1}\}\}$	NPC
$S_{3,6}$	$T^{\sigma}_{3,6} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon + \theta, a_{\sigma(i+2)} + 2\epsilon + \theta - w_2^i, b_{\sigma(i+1)} + 2\delta + 3\epsilon\}$ $w_2^i = \max\{0, b_{\sigma(i)} - \max\{3\delta + 3\epsilon + \theta, a_{\sigma(i+1)} - w_2^{i-1} - 2\delta - \epsilon + \theta\}\}$	NPC
$S_{3,9}$	$T^{\sigma}_{3,9} = \sum_{i=1}^{n} \max\{5\delta + 6\epsilon + \theta, a_{\sigma(i+2)} + 2\delta + 3\epsilon - w_2^i, b_{\sigma(i+1)} + \theta + 2\epsilon\}$ $w_2^i = \max\{0, b_{\sigma(i)} - \max\{5\delta + 4\epsilon, a_{\sigma(i+1)} - w_2^{i-1} + 2\delta + \epsilon - \theta\}\}$	NPC
$S_{3,10}$	$T^{\sigma}_{3,10} = \sum_{i=1}^{n} \max\{3\delta + 6\epsilon + 2\theta, a_{\sigma(i+2)} + \theta + 2\epsilon - w_2^i, b_{\sigma(i+1)} + \theta + 2\epsilon\}$ $w_2^i = \max\{0, b_{\sigma(i)} - \max\{3\delta + 4\epsilon + \theta, a_{\sigma(i+1)} - w_2^{i-1}\}\}$	NPC
$S_{3,11}$	$T^{\sigma}_{3,11} = \sum_{i=1}^{n} \max\{a_{\sigma(i+1)} + 3\delta + 6\epsilon + \theta, b_{\sigma(i)} + 2\epsilon + \theta\}$	P
$S_{3,16}$	$T^{\sigma}_{3,16} = \sum_{i=1}^{n} \max\{4\delta + 6\epsilon + \theta, a_{\sigma(i+2)} + 2\delta + 4\epsilon - w_2^i, b_{\sigma(i+1)} + 2\epsilon + \theta\}$ $w_2^i = \max\{0, b_{\sigma(i)} - \max\{4\delta + 4\epsilon, a_{\sigma(i+1)} - w_2^{i-1} + 2\delta + 2\epsilon - \theta\}\}$	NPC

Table 7.5. Complexity and Cycle-Time Expressions $T^{\sigma}_{u,v}$ for Problem $RF_2^2|(free,A,MP,S_{u,v} \in \mathcal{S})|\mu$ for Undominated CRM Sequences. P: Polynomial-Time, NPC: NP-Complete.

Let $\mathcal{S}_1 = \{S_{1,1}, S_{1,4}, S_{1,13}, S_{1,14}, S_{1,15}, S_{3,3}, S_{3,11}\}$ and $\mathcal{S}_2 = \{S_{3,4}, S_{3,5}, S_{3,6}, S_{3,9}, S_{3,16}\}$; thus, $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \{S_{3,10}\}$. We split the discussion into two cases. In Case 1, we consider only the sequences from set $\mathcal{S}_1$ and show that bound (7.28) is satisfied for any CRM sequence $S_{u,v} \in \mathcal{S}_1$. In Case 2, we derive an analogous result for sequences from set $\mathcal{S}_2$.

Case 1. We prove that bound (7.28) holds for any CRM sequence $S_{u,v}$ from set $\mathcal{S}_1$. Clearly, to prove the validity of (7.28), it is enough to show that $T^{\sigma}_{u,v} \geq T^{\sigma}_{3,10}$, $S_{u,v} \in \mathcal{S}_1$, for each schedule σ of parts. This inequality can be easily established by a straightforward comparison of the expression for $T^{\sigma}_{u,v}$ for each CRM sequence $S_{u,v} \in \mathcal{S}_1$ with the expression for $T^{\sigma}_{3,10}$.

We use the expression for estimating the value of $T^{\sigma}_{3,10}$ as given by (7.2)-(7.3). By exploiting the fact that $w^i_2 \geq 0$, we obtain

$$T^{\sigma}_{3,10} \leq \sum_{i=1}^{n} \max\{6\epsilon + 3\delta + 2\theta, a_{\sigma(i+1)} + 2\epsilon + \theta, b_{\sigma(i)} + 2\epsilon + \theta\}. \quad (7.29)$$

In Table 7.6, for each CRM sequence $S_{u,v} \in \mathcal{S}_1$, we present the expression for $T^{\sigma}_{u,v}$ paired with the bound on $T^{\sigma}_{3,10}$ obtained from (7.29). It is straightforward to check that our claim holds under condition (7.26).

Case 2. We now move to set $\mathcal{S}_2$. As in Case 1, we show that bound (7.28) holds for each CRM sequence $S_{u,v} \in \mathcal{S}_2$. The proof is based on an LP formulation of the problem of calculating the value of $T^{\sigma}_{u,v}$ subject to a given part schedule σ. For any CRM sequence $S_{u,v} \in \mathcal{S}_2 \cup S_{3,10}$, given a part schedule σ, we can calculate the value of $T^{\sigma}_{u,v}$ by solving the corresponding LP problem of the form (7.18)–(7.20) (see Section 7.2). In particular, for the CRM sequence $S_{3,10}$ we have

$$\begin{aligned}
\min T^{\sigma}_{3,10} &= 6n\epsilon + 3n\delta + 2n\theta + \sum_{i=1}^{n} \left(w^i_1 + w^i_2\right) && (7.30)\\
\text{s.t.} \quad w^i_1 + w^{i-2}_2 &\geq a_{\sigma(i)} - 4\epsilon - 3\delta - \theta, && (7.31)\\
w^{i+1}_1 + w^i_2 &\geq b_{\sigma(i)} - 4\epsilon - 3\delta - \theta, && (7.32)\\
w^i_1, w^i_2 &\geq 0, \quad i = 1, \ldots, n. && (7.33)
\end{aligned}$$

The proof is accomplished by considering sequences from set $\mathcal{S}_2$ one by one and comparing the corresponding value of $T^*_{u,v} = T^{\sigma^*_{u,v}}_{u,v}$ with $T^{\sigma^*_{u,v}}_{3,10}$. The argument goes as follows.

CRM	Cycle Time
$S_{1,1}$	$T^{\sigma}_{1,1} = 3n\epsilon + 2n\delta + \sum_{i=1}^{n} \max\{3\epsilon + 3\delta, a_{\sigma(i+1)} + 2\epsilon + \delta, b_{\sigma(i)}\}$ $T^{\sigma}_{3,10} \le 3n\epsilon + 2n\delta + \sum_{i=1}^{n} \max\{3\epsilon + \delta + 2\theta,$ $a_{\sigma(i+1)} + \theta - (\epsilon + 2\delta), b_{\sigma(i)} + \theta - (\epsilon + 2\delta)\}$
$S_{1,4}$	$T^{\sigma}_{1,4} = 4n\epsilon + 2n\delta + \sum_{i=1}^{n} \max\{a_{\sigma(i+1)}, b_{\sigma(i)} + 2\epsilon + 2\delta\}$ $T^{\sigma}_{3,10} \le 4n\epsilon + 2n\delta + \sum_{i=1}^{n} \max\{2\epsilon + \delta + 2\theta,$ $a_{\sigma(i+1)} + \theta - (2\epsilon + 2\delta), b_{\sigma(i)} + \theta - (2\epsilon + 2\delta)\}$
$S_{1,13}$	$T^{\sigma}_{1,13} = 6n\epsilon + 3n\delta + \sum_{i=1}^{n} (a_{\sigma(i+1)} + b_{\sigma(i)})$ $T^{\sigma}_{3,10} \le 6n\epsilon + 3n\delta + \sum_{i=1}^{n} \max\{2\theta,$ $a_{\sigma(i+1)} + \theta - (4\epsilon + 3\delta), b_{\sigma(i)} + \theta - (4\epsilon + 3\delta)\}$
$S_{1,14}$	$T^{\sigma}_{1,14} = 2n\epsilon + n\theta + \sum_{i=1}^{n} \max\{a_{\sigma(i+1)}, b_{\sigma(i)} + 4\epsilon + 3\delta\}$ $T^{\sigma}_{3,10} \le 2n\epsilon + n\theta + \sum_{i=1}^{n} \max\{4\epsilon + 3\delta + \theta, a_{\sigma(i+1)}, b_{\sigma(i)}\}$
$S_{1,15}$	$T^{\sigma}_{1,15} = 3n\epsilon + 2n\delta + \sum_{i=1}^{n} \max\{3\epsilon + 3\delta, a_{\sigma(i+1)}, b_{\sigma(i)} + 2\epsilon + \delta\}$ $T^{\sigma}_{3,10} \le 3n\epsilon + 2n\delta + \sum_{i=1}^{n} \max\{3\epsilon + \delta + 2\theta,$ $a_{\sigma(i+1)} + \theta - (\epsilon + 2\delta), b_{\sigma(i)} + \theta - (\epsilon + 2\delta)\}$
$S_{3,3}$	$T^{\sigma}_{3,3} = 4n\epsilon + 2n\delta + \sum_{i=1}^{n} \max\{a_{\sigma(i+1)} + 2\epsilon + 2\delta, b_{\sigma(i)}\}$ $T^{\sigma}_{3,10} \le 4n\epsilon + 2n\delta + \sum_{i=1}^{n} \max\{2\epsilon + \delta + 2\theta,$ $a_{\sigma(i+1)} + \theta - (2\epsilon + 2\delta), b_{\sigma(i)} + \theta - (2\epsilon + 2\delta)\}$
$S_{3,11}$	$T^{\sigma}_{3,11} = 2n\epsilon + n\theta + \sum_{i=1}^{n} \max\{a_{\sigma(i+1)} + 4\epsilon + 3\delta, b_{\sigma(i)}\}$ $T^{\sigma}_{3,10} \le 2n\epsilon + n\theta + \sum_{i=1}^{n} \max\{4\epsilon + 3\delta + \theta, a_{\sigma(i+1)}, b_{\sigma(i)}\}$

Table 7.6. Cycle-Time Expressions used in Case 1 of Theorem 7.6.

Let $S_{u,v}$ be a particular CRM sequence from set $\mathcal{S}_2$. For this particular sequence, let σ^* stand for an optimal part schedule $\sigma^*_{u,v}$. Given an optimal schedule σ^*, we denote by $\overline{w}^i_1$, $\overline{w}^i_2$, $i = 1, \ldots, n$, the solution to the LP problem (7.18)–(7.20) of finding the value $T^{\sigma^*}_{u,v}$. That is,

$$T^*_{u,v} = T^{\sigma^*}_{u,v} = Z_{u,v} + \sum_{i=1}^{n} \left(\overline{w}^i_1 + \overline{w}^i_2\right).$$

Next, given the values $\overline{w}^i_1$, $\overline{w}^i_2$, $i = 1, \ldots, n$, we define new values $\overline{\overline{w}}^i_1$ and $\overline{\overline{w}}^i_2$ that are guaranteed to satisfy the inequality constraints (7.31)–(7.33) in the LP problem (7.30)–(7.33) for estimating the value $T^{\sigma^*}_{3,10}$. Namely, one must have

$$\overline{\overline{w}}^i_1 + \overline{\overline{w}}^{i-2}_2 \;\ge\; a_{\sigma^*(i)} - 4\epsilon - 3\delta - \theta, \tag{7.34}$$

$$\overline{\overline{w}}^{i+1}_1 + \overline{\overline{w}}^i_2 \;\ge\; b_{\sigma^*(i)} - 4\epsilon - 3\delta - \theta, \tag{7.35}$$

$$\overline{\overline{w}}^i_1, \overline{\overline{w}}^i_2 \;\ge\; 0, \quad i = 1, \ldots, n. \tag{7.36}$$

Given the schedule σ^*, let $\overline{x}_1^i$, $\overline{x}_2^i$, $i = 1, \ldots, n$, be the solution to the LP problem (7.30)–(7.33) of finding the value $T_{3,10}^{\sigma^*}$. That is,

$$T_{3,10}^{\sigma^*} = 6n\epsilon + 3n\delta + 2n\theta + \sum_{i=1}^{n} \left(\overline{x}_1^i + \overline{x}_2^i\right).$$

Since the values of $\overline{\overline{w}}_1^i$ and $\overline{\overline{w}}_2^i$ satisfy the LP constraints (7.31)–(7.33), we have

$$\sum_{i=1}^{n} \left(\overline{\overline{w}}_1^i + \overline{\overline{w}}_2^i\right) \geq \sum_{i=1}^{n} \left(\overline{x}_1^i + \overline{x}_2^i\right). \tag{7.37}$$

Hence,

$$T_{3,10}^{*} \leq T_{3,10}^{\sigma^*} \leq 6n\epsilon + 3n\delta + 2n\theta + \sum_{i=1}^{n} \left(\overline{\overline{w}}_1^i + \overline{\overline{w}}_2^i\right). \tag{7.38}$$

Moreover, the values of $\overline{\overline{w}}_1^i$ and $\overline{\overline{w}}_2^i$ are defined in such a way that the following requirement holds:

$$6n\epsilon + 3n\delta + 2n\theta + \sum_{i=1}^{n} \left(\overline{\overline{w}}_1^i + \overline{\overline{w}}_2^i\right) = Z_{u,v} + \sum_{i=1}^{n} \left(\overline{w}_1^i + \overline{w}_2^i\right) = T_{u,v}^{\sigma^*}. \tag{7.39}$$

Then, (7.28) follows from (7.38) and (7.39). As an example, we show how this argument works for the CRM sequence $S_{3,4}$. An LP formulation of the problem of calculating the value of $T_{3,4}^{\sigma}$ subject to a given schedule σ follows:

$$\begin{aligned}
\min T_{3,4}^{\sigma} &= 6n\epsilon + 4n\delta + n\theta + \sum_{i=1}^{n} \left(w_1^i + w_2^i\right) \\
\text{s.t.} \quad w_1^i + w_2^{i-2} &\geq a_{\sigma(i)} - 4\epsilon - 4\delta, \\
w_1^{i+1} + w_2^i &\geq b_{\sigma(i)} - 2\epsilon - 2\delta - \theta, \\
w_1^i, w_2^i &\geq 0, \quad i = 1, \ldots, n.
\end{aligned}$$

Let $\overline{w}_1^i$, $\overline{w}_2^i$, $i = 1, \ldots, n$, be a solution to the above LP problem, so that

$$T_{3,4}^{*} = T_{3,4}^{\sigma^*} = 6n\epsilon + 4n\delta + n\theta + \sum_{i=1}^{n} \left(\overline{w}_1^i + \overline{w}_2^i\right), \tag{7.40}$$

$$\begin{aligned}
\overline{w}_1^i + \overline{w}_2^{i-2} &\geq a_{\sigma^*(i)} - 4\epsilon - 4\delta, \quad \overline{w}_1^{i+1} + \overline{w}_2^i \geq b_{\sigma^*(i)} - 2\epsilon - 2\delta - \theta, \\
&\overline{w}_1^i, \overline{w}_2^i \geq 0, \ i = 1, \ldots, n.
\end{aligned} \tag{7.41}$$

We define the values of $\overline{\overline{w}}_1^i$ and $\overline{\overline{w}}_2^i$ as follows:

$$\overline{\overline{w}}_1^i = \overline{w}_1^i + \delta - \theta; \qquad \overline{\overline{w}}_2^i = \overline{w}_2^i.$$

From (7.41), we obtain

$$\begin{aligned} \overline{\overline{w}}_1^i + \overline{\overline{w}}_2^{i-2} &= \overline{w}_1^i + \delta - \theta + \overline{w}_2^{i-2} \geq a_{\sigma^*(i)} - 4\epsilon - 3\delta - \theta, \\ \overline{\overline{w}}_1^{i+1} + \overline{\overline{w}}_2^i &= \overline{w}_1^{i+1} + \delta - \theta + \overline{w}_2^i \geq b_{\sigma^*(i)} - 2\epsilon - \delta - 2\theta. \end{aligned}$$

Thus, for the values of $\overline{\overline{w}}_1^i$ and $\overline{\overline{w}}_2^i$, the inequality constraints (7.34)–(7.36) in the LP problem of estimating the value $T_{3,10}^{\sigma^*}$ are satisfied as a result of (7.26). Hence, we have (7.37), and then (7.38). Finally, we derive

$$\sum_{i=1}^{n} \left(\overline{\overline{w}}_1^i + \overline{\overline{w}}_2^i \right) = \sum_{i=1}^{n} \left(\overline{w}_1^i + \delta - \theta + \overline{w}_2^i \right) = n\delta - n\theta + \sum_{i=1}^{n} \left(\overline{w}_1^i + \overline{w}_2^i \right),$$

and then

$$6n\epsilon + 3n\delta + 2n\theta + \sum_{i=1}^{n} \left(\overline{\overline{w}}_1^i + \overline{\overline{w}}_2^i \right) = 6n\epsilon + 4n\delta + n\theta + \sum_{i=1}^{n} \left(\overline{w}_1^i + \overline{w}_2^i \right) = T_{3,4}^{\sigma^*}.$$

Thus, (7.39) holds, and so does the desired bound (7.28).

CRM $S_{u,v}$	$S_{3,4}$	$S_{3,5}$	$S_{3,6}$	$S_{3,9}$	$S_{3,16}$
$\overline{\overline{w}}_1^i$	$\overline{w}_1^i + \delta - \theta$	$\overline{w}_1^i + 3\delta - 2\theta$	$\overline{w}_1^i + 2\delta - \theta$	$\overline{w}_1^i + 2\delta - \theta$	$\overline{w}_1^i + \delta - \theta$
$\overline{\overline{w}}_2^i$	$\overline{w}_2^i$	$\overline{w}_2^i$	$\overline{w}_2^i$	$\overline{w}_2^i$	$\overline{w}_2^i$

Table 7.7. Robot Waiting Times.

The proof for other sequences from set $\mathcal{S}_2$ is analogous. The values of $Z_{u,v}$, $\gamma_{u,v}^a$, and $\gamma_{u,v}^b$ for all the remaining sequences $S_{u,v} \in \mathcal{S}_2$ in the corresponding LP problem of finding the value $T_{u,v}^{\sigma}$ are as given in Table 7.4. The transformation to derive the values $\overline{\overline{w}}_1^i$ and $\overline{\overline{w}}_2^i$ from $\overline{w}_1^i$ and $\overline{w}_2^i$, respectively, is defined in Table 7.7. ∎

7.4 A Heuristic for Specific CRM Sequences

As shown in Section 7.2, the part scheduling problems $RF_2^2|(free, A, MP, S_{u,v})|\mu$ associated with the CRM sequences $S_{u,v} \in \widetilde{\mathcal{S}} = \{S_{3,4}, S_{3,5},$

$S_{3,6}$, $S_{3,9}$, $S_{3,10}$, $S_{3,16}\}$ are strongly NP-hard. Below we describe a heuristic procedure to solve these problems. Note that the root cause of the intractability of these problems is the presence of the robot waiting-time terms w_2^i, $i = 1, 2, \ldots, n$, in their cycle time expressions (see Table 7.5). The idea behind the heuristic is easy to explain: by ignoring the w_2^i-terms in the cycle time expressions, we can reduce the problem to a special case of the traveling salesman problem that is solvable by the Gilmore-Gomory algorithm. The part schedule thus obtained is used as a heuristic solution to the original problem.

Given a robot move sequence $S_{u,v} \in \widetilde{\mathcal{S}}$, define the values $Z_{u,v}$, $\gamma_{u,v}^a$, and $\gamma_{u,v}^b$ as in Table 7.4. Furthermore, given parts processing times a_k and b_k, $P_k \in \mathcal{P}$, define the values $\overline{a}_k$ and $\overline{b}_k$ as follows:

$$\overline{a}_k = \max\left\{0, a_k - \gamma_{u,v}^a\right\}, \quad \overline{b}_k = \max\left\{0, b_k - \gamma_{u,v}^b\right\}, \quad P_k \in \mathcal{P}. \quad (7.42)$$

Algorithm Hard-CRM

Input: An instance of $RF_2^2|(free,A,MP,S_{u,v})|\mu$ with a specified $S_{u,v} \in \widetilde{\mathcal{S}}$, where $\widetilde{\mathcal{S}} = \{S_{3,4}, S_{3,5}, S_{3,6}, S_{3,9}, S_{3,10}, S_{3,16}\}$.

Output: A schedule $\rho^* = (\rho^*(1), \rho^*(2), \ldots, \rho^*(n))$ of parts $P_k \in \mathcal{P}$ and $T_{u,v}^{\rho^*}$.

Step 1: Find values $\gamma_{u,v}^a$ and $\gamma_{u,v}^b$ defined in Table 7.4.

For each part $P_k \in \mathcal{P}$, find values $\overline{a}_k = \max\left\{0, a_k - \gamma_{u,v}^a\right\}$, $\overline{b}_k = \max\left\{0, b_k - \gamma_{u,v}^b\right\}$.

Step 2: Use the Gilmore-Gomory algorithm to obtain a schedule ρ^* that minimizes the function $\sum_i \max\left\{\overline{a}_{\rho^*(i+1)}, \overline{b}_{\rho^*(i)}\right\}$. Find $T_{u,v}^{\rho^*}$. Stop.

The running time of algorithm Hard-CRM is $O(n \log n)$. In the next section, we show that the worst-case bound of algorithm Hard-CRM is 3/2. This suggests that the heuristic is a promising tool for a good approximate solution. Later, in Section 7.5, we use Hard-CRM to devise an approximation algorithm for the general problem over all CRM sequences.

7.4.1 A Performance Bound for Heuristic Hard-CRM

In this section, we derive a performance bound for algorithm Hard-CRM. The expressions for the cycle time of an MPS cycle in a two-

machine dual-gripper robotic cell, subject to a given part schedule $\sigma = (\sigma(1), \ldots, \sigma(n))$ and a specified robot move sequence, were derived in Section 7.2. Below we present such expressions for $S_{q,r} \in \widetilde{\mathcal{S}} = \{S_{3,4}, S_{3,5}, S_{3,6}, S_{3,9}, S_{3,10}, S_{3,16}\}$ unified under a generalized framework. It is then an easy exercise to verify that for $RF_2^2|(free, A, MP, S_{q,r} \in \widetilde{\mathcal{S}})|C_t$, the cycle time $T_{q,r}^{\sigma}$ for part schedule σ admits a representation of the general form (see Section 7.2, (7.4)–(7.7)). For simplicity, we use u_i, v_i for w_i^1, w_i^2, respectively:

$$T_{q,r}^{\sigma} = Z_{q,r} + \sum_{i=1}^{n} \{u_i + v_i\}, \tag{7.43}$$

$$\begin{aligned} u_i &= \max\left\{0, \overline{a}_{\sigma(i)} - v_{i-2}\right\}, \quad v_i = \max\left\{0, \overline{b}_{\sigma(i)} - u_{i+1}\right\}, \\ & u_i = u_{i\pm n}, \quad v_i = v_{i\pm n}. \end{aligned} \tag{7.44}$$

By simple algebraic transformations, the expressions (7.43)-(7.44) can be modified as

$$T_{q,r}^{\sigma} = Z_{q,r} + \sum_{i=1}^{n} \max\left\{\overline{a}_{\sigma(i+2)} - v_i, \overline{b}_{\sigma(i+1)}\right\}, \tag{7.45}$$

$$v_i = \max\left\{0, \overline{b}_{\sigma(i)} - \max\left\{0, \overline{a}_{\sigma(i+1)} - v_{i-1}\right\}\right\}, \quad v_i = v_{i\pm n}. \tag{7.46}$$

Note that the cycle time $T_{q,r}^{\sigma}$ consists of the robot activity time $Z_{q,r}$ (load, unload, move, and switch times of the robot) and robot wait times u_i, v_i at machines M_1, M_2, respectively, before parts are unloaded from those machines. Moreover, $\gamma_{q,r}^{a}$ (resp., $\gamma_{q,r}^{b}$) denotes the robot activity time between the moment the robot loads a part on M_1 (resp., M_2) and the moment it returns to M_1 (resp., M_2) to unload the part.

The formulation given in (7.43) and (7.44) is easily modified to yield the formulation given below in Lemma 7.6.

LEMMA 7.6 *For $RF_2^2|(free, A, MP, S_{q,r} \in \widetilde{\mathcal{S}})|\mu$, the problem of calculating the value of $T_{q,r}^{\sigma}$ subject to a given schedule σ of parts admits the following LP formulation:*

$$\min \quad T_{q,r}^{\sigma} = Z_{q,r} + \sum_{i=1}^{n} (u_i + v_i) \tag{7.47}$$

$$s.t. \quad u_i + v_{i-2} \geq \overline{a}_{\sigma(i)}, \quad i = 1, \ldots, n, \tag{7.48}$$

$$u_{i+1} + v_i \geq \overline{b}_{\sigma(i)}, \quad i = 1, \ldots, n, \tag{7.49}$$

$$u_i, v_i \geq 0, \quad u_i = u_{i\pm n}, \quad v_i = v_{i\pm n}. \tag{7.50}$$

Proof. The proof follows from (7.43) and (7.44). ∎

We are now ready to prove the results that are required to establish a bound for algorithm Hard-CRM. In what follows, we exploit known results from no-wait flow shop scheduling. More precisely, we relate our problem to $F_2|no\text{-}wait|C_t$. The latter problem has as input a set of n jobs, with each job i associated with two numerical parameters $\overline{a}_i$ and $\overline{b}_i$. The objective is to find a job schedule ρ that minimizes $\sum_{i=1}^{n} \max\left\{\overline{a}_{\rho(i+1)}, \overline{b}_{\rho(i)}\right\}$. Problem $F_2|no\text{-}wait|C_t$ is known to be solvable in time $O(n \log n)$ by the Gilmore-Gomory algorithm (see Appendix B).

Our main result in this section is a proof that for $RF_2^2|(free,A,MP, S_{q,r} \in \widetilde{\mathcal{S}})|\mu$, algorithm Hard-CRM finds an approximate solution that is at most 3/2 times the optimal value. Theorem 7.7 establishes the connection between $F_2|no\text{-}wait|C_t$ and $RF_2^2|(free,A,MP, S_{q,r} \in \widetilde{\mathcal{S}})|\mu$.

THEOREM 7.7 *For any schedule σ of parts, $RF_2^2|(free,A,MP,S_{q,r} \in \widetilde{\mathcal{S}})|\mu$ has a schedule ρ such that*

$$\sum_{i=1}^{n} \max\left\{\overline{a}_{\rho(i+1)}, \overline{b}_{\rho(i)}\right\} \leq \frac{3}{2}\left(T_{q,r}^{\sigma} - Z_{q,r}\right).$$

Proof. Let $\sigma = (\sigma(1),\ \sigma(2),\ \ldots,\ \sigma(n))$, $n \geq 3$, be an arbitrary schedule of parts. By re-indexing if necessary, we may assume without loss of generality that $\sigma(i) = i$, $i = 1, \ldots, n$, so that

$$\sigma = (1,\ 2,\ \ldots,\ n). \tag{7.51}$$

Furthermore, for the sake of simplicity, in what follows we will use the notation $\sum$ to denote $\sum_{i=1}^{n}$, unless explicitly stated otherwise.

Given the schedule σ, let $\{u_i\}_{i=1,\ldots,n}$, $\{v_i\}_{i=1,\ldots,n}$ be any feasible solution of LP problem (7.47)–(7.50), with T_u^{σ} being the corresponding value of the objective function (7.47). Thus, we aim to prove that there always exists a schedule ρ such that

$$\sum \max\left\{\overline{a}_{\rho(i+1)}, \overline{b}_{\rho(i)}\right\} \leq \frac{3}{2}\sum (u_i + v_i). \tag{7.52}$$

We note that the constraints (7.48)–(7.49) of the LP, if rewritten subject to (7.51), take the form

$$u_i + v_{i-2} \geq \overline{a}_i, \quad i = 1, \ldots, n, \tag{7.53}$$

$$u_{i+1} + v_i \geq \overline{b}_i, \quad i = 1, \ldots, n. \tag{7.54}$$

Furthermore, the inequalities (7.53)–(7.54) immediately yield the following relationships, which we will exploit later in the proof:

$$v_i + \max\{u_{i+1}, u_{i+2}\} \geq \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\}, \tag{7.55}$$

$$v_i + \min\{u_{i+1}, u_{i+2}\} \geq \min\left\{\overline{a}_{i+2}, \overline{b}_i\right\}, \tag{7.56}$$

$$u_{i+1} + \max\{v_{i-1}, v_i\} \geq \max\left\{\overline{a}_{i+1}, \overline{b}_i\right\}, \tag{7.57}$$

$$u_{i+1} + \min\{v_{i-1}, v_i\} \geq \min\left\{\overline{a}_{i+1}, \overline{b}_i\right\}. \tag{7.58}$$

We then obtain (using (7.57))

$$\begin{aligned} \sum (u_i + v_i) &= \sum u_i + \sum \max\{v_{i-1}, v_i\} + \sum \min\{v_{i-1}, v_i\} - \sum v_i \\ &\geq \sum \max\left\{\overline{a}_{i+1}, \overline{b}_i\right\} + \sum \min\{v_{i-1}, v_i\} - \sum v_i. \end{aligned}$$

Hence,

$$\sum \max\left\{\overline{a}_{i+1}, \overline{b}_i\right\} \leq \sum u_i + 2\sum v_i - \sum \min\{v_{i-1}, v_i\}.$$

If $\sum v_i - \sum \min\{v_{i-1}, v_i\} \leq \frac{1}{2}\sum (u_i + v_i)$, then we set $\rho = \sigma$, and (7.52) is satisfied. Thus, in the remainder of the proof, we assume that

$$\frac{1}{2}\sum v_i > \frac{1}{2}\sum u_i + \sum \min\{v_{i-1}, v_i\}. \tag{7.59}$$

Furthermore, we obtain (from (7.55))

$$\begin{aligned} \sum (u_i + v_i) &= \sum v_i + \sum \max\{u_{i-1}, u_i\} + \sum \min\{u_{i-1}, u_i\} - \sum u_i \\ &\geq \sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} + \sum \min\{u_{i-1}, u_i\} - \sum u_i. \end{aligned}$$

Hence,

$$\sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} \leq 2\sum u_i + \sum v_i - \sum \min\{u_{i-1}, u_i\}.$$

In view of (7.59), this leads to

$$\begin{aligned} \sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} &< \frac{3}{2}\sum (u_i + v_i) - \sum \min\{u_{i-1}, u_i\} \\ &\quad - \sum \min\{v_{i-1}, v_i\}. \end{aligned} \tag{7.60}$$

For the remainder of the proof, we distinguish between two cases depending on the value of n.

Case 1. n is odd: In this case, we set $\rho = (\sigma(1),\ \sigma(3),\ \ldots,\ \sigma(n),\ \sigma(2),\ \sigma(4),\ \ldots,\ \sigma(n-1))$ so that

$$\sum \max\left\{\overline{a}_{\rho(i+1)}, \overline{b}_{\rho(i)}\right\} = \sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\},$$

and the result follows immediately from (7.60).

Case 2. n is even, i.e., $n = 2k$, $k \geq 2$: In contrast to Case 1, we face the situation in which the function $\sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\}$ cannot be immediately associated with the desired schedule ρ. Let $I = \{1,\ 3,\ \ldots,\ n-1\}$ and $J = \{2,\ 4,\ \ldots,\ n\}$. Furthermore, let φ and ψ be the subschedules of the jobs defined, respectively, on the sets I and J:

$$\varphi = (1,\ 3,\ \ldots,\ n-1)\,,\quad \psi = (2,\ 4,\ \ldots,\ n)\,.$$

We then have

$$\sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} = \sum_{i=1}^{k} \max\left\{\overline{a}_{\varphi(i+1)}, \overline{b}_{\varphi(i)}\right\} + \sum_{i=1}^{k} \max\left\{\overline{a}_{\psi(i+1)}, \overline{b}_{\psi(i)}\right\}. \tag{7.61}$$

We now aim to combine these two subschedules φ and ψ to obtain the resulting schedule ρ that delivers the bound (7.52). More precisely, we will identify two indices x and y, where $x \in I$, $y \in J$, and construct the desired schedule ρ by joining φ and ψ as follows:

$$\begin{aligned}\rho \;=\; & (1,\ 3,\ \ldots,\ x,\ y+2,\ y+4,\ \ldots,\ n,\ 2,\ 4,\ \ldots,\ y,\ x+2,\\ & x+4,\ \ldots,\ n-1).\end{aligned} \tag{7.62}$$

We denote by Δ the amount by which the cost of ρ exceeds $\sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\}$, i.e.,

$$\sum \max\left\{\overline{a}_{\rho(i+1)}, \overline{b}_{\rho(i)}\right\} = \sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} + \Delta. \tag{7.63}$$

Taking into account the structure of the schedule ρ given by (7.62), the latter relation yields (using (7.61)

$$\begin{aligned}
\Delta &= \sum \max\left\{\overline{a}_{\rho(i+1)}, \overline{b}_{\rho(i)}\right\} - \sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} \\
&= \sum \max\left\{\overline{a}_{\rho(i+1)}, \overline{b}_{\rho(i)}\right\} - \left(\sum_{i=1}^{k} \max\left\{\overline{a}_{\varphi(i+1)}, \overline{b}_{\varphi(i)}\right\}\right. \\
&\qquad \left. + \sum_{i=1}^{k} \max\left\{\overline{a}_{\psi(i+1)}, \overline{b}_{\psi(i)}\right\}\right) \\
&= \max\left\{\overline{a}_{x+2}, \overline{b}_y\right\} + \max\left\{\overline{a}_{y+2}, \overline{b}_x\right\} \\
&\qquad - \max\left\{\overline{a}_{x+2}, \overline{b}_x\right\} - \max\left\{\overline{a}_{y+2}, \overline{b}_y\right\},
\end{aligned}$$

so that

$$\Delta = \max\left\{\overline{a}_{x+2}, \overline{b}_y\right\} + \max\left\{\overline{a}_{y+2}, \overline{b}_x\right\} - \max\left\{\overline{a}_{x+2}, \overline{b}_x\right\} - \max\left\{\overline{a}_{y+2}, \overline{b}_y\right\}. \tag{7.64}$$

The above relations are illustrated in Figure 7.9.

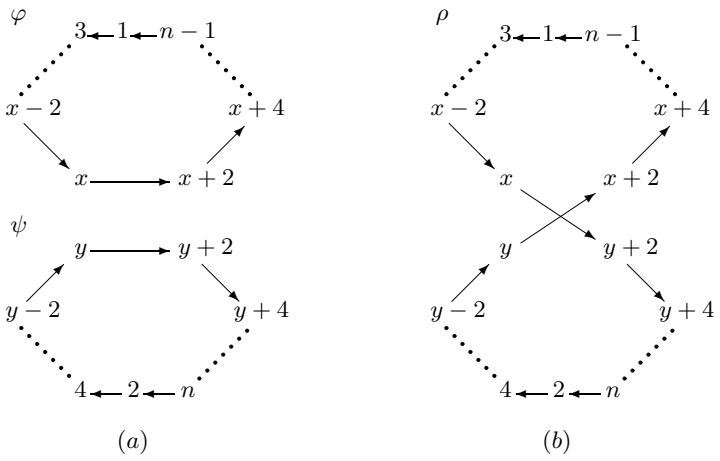


Figure 7.9. (a) Subschedules φ and ψ (b) Sequence ρ.

The remainder of the proof is split into two cases.

Case 2.1. Suppose there exists a pair of indices i and j, $i \in I$, $j \in J$, such that

$$\min\left\{\max\{\overline{a}_{i+2}, \overline{b}_i\},\ \max\{\overline{a}_{j+2}, \overline{b}_j\}\right\} \geq \max\{\min\{\overline{a}_{i+2}, \overline{b}_i\}, \min\{\overline{a}_{j+2}, \overline{b}_j\}\}. \quad (7.65)$$

In this case, we identify $x \in I$ and $y \in J$, for which (7.65) holds with $i = x$ and $j = y$, and construct the schedule ρ as specified by (7.62). The condition (7.65) yields

$$\max\left\{\overline{a}_{x+2}, \overline{b}_y\right\} + \max\left\{\overline{a}_{y+2}, \overline{b}_x\right\} \leq \max\left\{\overline{a}_{x+2}, \overline{b}_x\right\} + \max\left\{\overline{a}_{y+2}, \overline{b}_y\right\}.$$

Thus, by (7.64), we obtain $\Delta \leq 0$. Combining $\Delta \leq 0$ with (7.60) and (7.63), we derive directly the desired bound (7.52).

Case 2.2. Suppose the condition (7.65) does not hold for any pair of indices i and j, $i \in I$, $j \in J$. This implies that

$$\min\{\max\{\overline{a}_{i+2}, \overline{b}_i\},\ \max\{\overline{a}_{j+2}, \overline{b}_j\}\} < \max\{\min\{\overline{a}_{i+2}, \overline{b}_i\}, \min\{\overline{a}_{j+2}, \overline{b}_j\}\}, \quad (7.66)$$

for any pair of indices i and j, $i \in I$, $j \in J$.

By reindexing (or renaming the sets I and J), we may further assume, without loss of generality, that

$$\min_{i \in I}\left\{\min\left\{\overline{a}_{i+2}, \overline{b}_i\right\}\right\} > \min_{i \in J}\left\{\min\left\{\overline{a}_{i+2}, \overline{b}_i\right\}\right\}. \quad (7.67)$$

We may then identify indices x and y, $x \in I$, $y \in J$, such that

$$\min\left\{\overline{a}_{x+2}, \overline{b}_x\right\} = \min_{i \in I}\left\{\min\left\{\overline{a}_{i+2}, \overline{b}_i\right\}\right\}, \quad (7.68)$$

$$\max\left\{\overline{a}_{y+2}, \overline{b}_y\right\} = \max_{i \in J}\left\{\max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} \middle| \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} < \min\left\{\overline{a}_{x+2}, \overline{b}_x\right\}\right\}. \quad (7.69)$$

We note that due to (7.66) and (7.67), such indices x and y, $x \in I$, $y \in J$, always exist. Having identified the indices x and y specified by (7.68) and (7.69), we form the schedule ρ as defined by (7.62).

From (7.69), we have $\max\left\{\overline{a}_{x+2}, \overline{b}_y\right\} = \overline{a}_{x+2}$ and $\max\left\{\overline{a}_{y+2}, \overline{b}_x\right\} = \overline{b}_x$. Combining the latter with (7.64) yields $\Delta = \overline{a}_{x+2} + \overline{b}_x - \max\left\{\overline{a}_{x+2}, \overline{b}_x\right\} - \max\left\{\overline{a}_{y+2}, \overline{b}_y\right\}$, or, equivalently,

$$\Delta = \min\left\{\overline{a}_{x+2}, \overline{b}_x\right\} - \max\left\{\overline{a}_{y+2}, \overline{b}_y\right\}. \quad (7.70)$$

We will now prove that the theorem holds for the schedule ρ. The proof is by contradiction.

Suppose that (7.52) does not hold. By (7.63), this implies

$$\sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} + \Delta > \frac{3}{2}\sum (u_i + v_i)\,. \tag{7.71}$$

We then derive

$$\begin{aligned}
\Delta &> \frac{3}{2}\sum u_i + \frac{1}{2}\sum v_i + \left(\sum v_i - \sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\}\right) \\
&\geq \frac{3}{2}\sum u_i + \frac{1}{2}\sum v_i - \sum \max\{u_i, u_{i+1}\} \text{ [by (7.55)]} \\
&= -\frac{1}{2}\sum u_i + \frac{1}{2}\sum v_i + \sum \min\{u_i, u_{i+1}\} \\
&> \sum \min\{v_i, v_{i+1}\} + \sum \min\{u_i, u_{i+1}\}\,. \text{ [by (7.59)]}
\end{aligned}$$

Thus, we have

$$\Delta > \sum \min\{v_i, v_{i+1}\} + \sum \min\{u_i, u_{i+1}\}\,. \tag{7.72}$$

Combining (7.72) with (7.70), we get

$$\begin{aligned}
\sum \min\{v_i, v_{i+1}\} \;+\; &\sum \min\{u_i, u_{i+1}\} \\
&< \min\left\{\overline{a}_{x+2}, \overline{b}_x\right\} - \max\left\{\overline{a}_{y+2}, \overline{b}_y\right\} \\
&\leq \min\left\{\overline{a}_{x+2}, \overline{b}_x\right\} \\
&= \min_{i\in I}\left\{\min\left\{\overline{a}_{i+2}, \overline{b}_i\right\}\right\} \text{ [by (7.68)]} \\
&\leq \min_{i\in I}\{v_i + \min\{u_{i+1}, u_{i+2}\}\} \text{ [by (7.56)]} \\
&\leq \min_{i\in I} v_i + \sum \min\{u_i, u_{i+1}\}\,.
\end{aligned}$$

Therefore, we obtain

$$\sum \min\{v_i, v_{i+1}\} < \min_{i\in I} v_i. \tag{7.73}$$

(7.73) implies that for *any* index $i \in I$, one must have $\min\{v_{i-1}, v_i\} = v_{i-1}$, as well as $\min\{v_i, v_{i+1}\} = v_{i+1}$. Consequently,

$$\begin{aligned}
\sum \min\{v_i, v_{i+1}\} &= \sum_{i\in I} \min\{v_{i-1}, v_i\} + \sum_{i\in I} \min\{v_i, v_{i+1}\} \\
&= 2\sum_{i\in J} v_i. \qquad (7.74)
\end{aligned}$$

Hence, (7.72) yields

$$\begin{aligned}
\Delta &> \sum \min\{v_i, v_{i+1}\} + \sum \min\{u_i, u_{i+1}\} \\
&= 2\sum_{i\in J} v_i + \sum \min\{u_i, u_{i+1}\} \text{ [by (7.74)]} \\
&\geq \sum_{i\in J} v_i + \sum_{i\in J} \min\{u_{i+1}, u_{i+2}\} \\
&\geq \sum_{i\in J} \min\left\{\overline{a}_{i+2}, \overline{b}_i\right\} \geq \max_{i\in J}\left\{\min\left\{\overline{a}_{i+2}, \overline{b}_i\right\}\right\}. \text{ [by (7.56)]}
\end{aligned}$$

Combining the above result with (7.70), we obtain

$$\begin{aligned}
\max_{i\in J}\left\{\min\left\{\overline{a}_{i+2}, \overline{b}_i\right\}\right\} &< \Delta = \min\left\{\overline{a}_{x+2}, \overline{b}_x\right\} - \max\left\{\overline{a}_{y+2}, \overline{b}_y\right\} \\
&\leq \min\left\{\overline{a}_{x+2}, \overline{b}_x\right\} = \min_{i\in I}\left\{\min\left\{\overline{a}_{i+2}, \overline{b}_i\right\}\right\}. \\
&\qquad \text{[by (7.68)]}
\end{aligned}$$

Thus, we have

$$\min_{i\in I}\left\{\min\left\{\overline{a}_{i+2}, \overline{b}_i\right\}\right\} > \max_{i\in J}\left\{\min\left\{\overline{a}_{i+2}, \overline{b}_i\right\}\right\}.$$

It then follows from (7.66) that for any pair of indices i and j, $i \in I$, $j \in J$, one must have $\min\left\{\overline{a}_{i+2}, \overline{b}_i\right\} > \max\left\{\overline{a}_{j+2}, \overline{b}_j\right\}$. Therefore, by (7.69),

$$\max\left\{\overline{a}_{y+2}, \overline{b}_y\right\} = \max_{i\in J}\left\{\max\left\{\overline{a}_{i+2}, \overline{b}_i\right\}\right\}, \tag{7.75}$$

and, therefore,

$$\begin{aligned}
\Delta &= \min\left\{\overline{a}_{x+2}, \overline{b}_x\right\} - \max\left\{\overline{a}_{y+2}, \overline{b}_y\right\} \text{ [by (7.70)]} \\
&\leq \min_{i\in I}\left\{\min\left\{\overline{a}_{i+2}, \overline{b}_i\right\}\right\} - \max_{i\in J}\left\{\max\left\{\overline{a}_{i+2}, \overline{b}_i\right\}\right\} \text{ [by (7.68), (7.75)]} \\
&\leq \frac{1}{k}\sum_{i\in I} \min\left\{\overline{a}_{i+2}, \overline{b}_i\right\} - \frac{1}{k}\sum_{i\in J} \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\}.
\end{aligned}$$

Thus, we have

$$\Delta \leq \frac{1}{k}\sum_{i\in I} \min\left\{\overline{a}_{i+2}, \overline{b}_i\right\} - \frac{1}{k}\sum_{i\in J} \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\}. \tag{7.76}$$

Using the previously obtained relations, we then derive

$$\frac{3}{2}\sum (u_i + v_i) < \sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} + \Delta \quad \text{[by (7.71)]}$$

$$
\begin{aligned}
&\leq \sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} + \frac{1}{k}\sum_{i\in I} \min\left\{\overline{a}_{i+2}, \overline{b}_i\right\} - \frac{1}{k}\sum_{i\in J} \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{[by (7.76)]} \\
&= \frac{k-1}{k}\sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} + \frac{1}{k}\sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} \\
&\qquad\qquad + \frac{1}{k}\sum_{i\in I} \min\left\{\overline{a}_{i+2}, \overline{b}_i\right\} - \frac{1}{k}\sum_{i\in J} \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} \\
&= \frac{k-1}{k}\sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} + \frac{1}{k}\sum_{i\in I} \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} \\
&\qquad\qquad + \frac{1}{k}\sum_{i\in I} \min\left\{\overline{a}_{i+2}, \overline{b}_i\right\} \\
&= \frac{k-1}{k}\sum \max\left\{\overline{a}_{i+2}, \overline{b}_i\right\} + \frac{1}{k}\sum_{i\in I} \left(\overline{a}_{i+2} + \overline{b}_i\right) \\
&\leq \frac{k-1}{k}\sum \left(v_i + \max\left\{u_{i+1}, u_{i+2}\right\}\right) + \frac{1}{k}\sum_{i\in I} \left(2v_i + u_{i+1} + u_{i+2}\right) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{[by (7.53), (7.54), (7.55)]} \\
&= \frac{k-1}{k}\sum v_i + \frac{2}{k}\sum_{i\in I} v_i + \frac{k-1}{k}\sum \max\left\{u_{i+1}, u_{i+2}\right\} + \frac{1}{k}\sum u_i \\
&\leq \left(\frac{k-1}{k} + \frac{2}{k}\right)\sum v_i + \left(\frac{2\,(k-1)}{k} + \frac{1}{k}\right)\sum u_i \\
&= \frac{k+1}{k}\sum v_i + \frac{2k-1}{k}\sum u_i.
\end{aligned}
$$

Combining the head and the tail of the latter chain of inequalities, we obtain

$$\frac{k-2}{2k}\sum v_i < \frac{k-2}{2k}\sum u_i.$$

This is a contradiction to (7.59) and n even with $n = 2k$, $k \geq 2$, which completes the proof. ∎

We now provide a numerical example to illustrate algorithm Hard-CRM.

EXAMPLE 7.1 Consider an instance of $RF_2^2|(free,A,MP,\ S_{q,r} \in \widetilde{\mathcal{S}}\ )|\mu$. Let $S_{q,r} = S_{3,10}$, and let an MPS consist of five parts, i.e., $n = 5$ and $\mathcal{P} = \{P_1, P_2, P_3, P_4, P_5\}$. Furthermore, let $\theta = 1$, $\epsilon = 2$, and $\delta = 3$. The processing times a_k and b_k of parts $P_k \in \mathcal{P}$ are given in Table 7.8.

We illustrate the steps of algorithm Hard-CRM for this data. For $S_{3,10}$, we find the values of $Z_{3,10}$, $\gamma^a_{3,10}$, and $\gamma^b_{3,10}$, as defined in Table 7.4:

$$\begin{aligned} Z_{3,10} &= 6n\epsilon + 3n\delta + 2n\theta = 115, \\ \gamma^a_{3,10} &= \gamma^b_{3,10} = 4\epsilon + 3\delta + \theta = 18. \end{aligned}$$

Step 1: Using equation (7.42), we calculate $\overline{a}_k = \max\{0, a_k - \gamma^a_{3,10}\}$ and $\overline{b}_k = \max\{0, b_k - \gamma^b_{3,10}\}$, $P_k \in \mathcal{P}$, as shown in Table 7.9.

k	1	2	3	4	5
a_k	27	24	26	22	20
b_k	21	23	23	24	25

Table 7.8. Processing Times of Parts in $\mathcal{P} = \{P_1, P_2, P_3, P_4, P_5\}$.

Step 2: We use the Gilmore-Gomory algorithm to find a schedule ρ^* that minimizes the functional $\sum_{i=1}^{5} \max\{\overline{a}_{\rho(i+1)}, \overline{b}_{\rho(i)}\}$. The algorithm starts by finding a simple matching of parameters $\overline{a}_k$ and $\overline{b}_k$, as shown in Table 7.10, by sequencing them in non-decreasing order: $\overline{b}_1 \le \overline{b}_2 \le \overline{b}_3 \le \overline{b}_4 \le \overline{b}_5$ and $\overline{a}_{\rho(1)} \le \overline{a}_{\rho(2)} \le \overline{a}_{\rho(3)} \le \overline{a}_{\rho(4)} \le \overline{a}_{\rho(5)}$.

k	1	2	3	4	5
$\overline{a}_k$	9	6	8	4	2
$\overline{b}_k$	3	5	5	6	7

Table 7.9. Values of $\overline{a}_k$ and $\overline{b}_k$, $k = 1, \ldots, 5$.

A TSP tour is constructed from Table 7.10 by pairing city k (origin) with city $\rho(k)$ (destination), $k = 1, \ldots, 5$. If we obtain a valid tour straight away, the algorithm stops. Otherwise there are subtours, and further work is required. The latter is our case as we have two subtours:

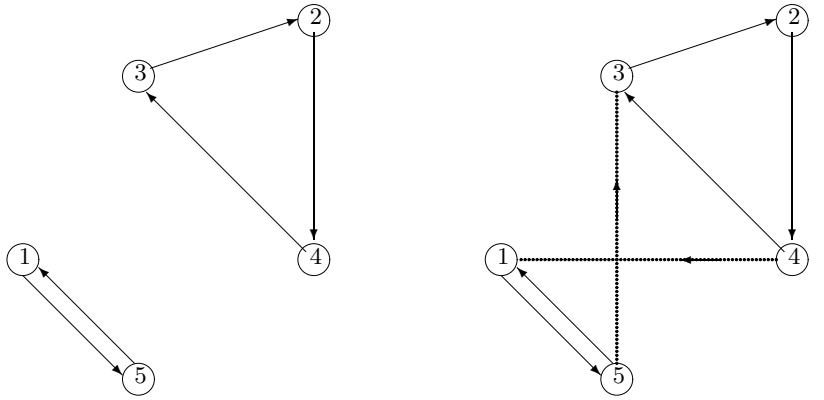
$$\begin{aligned} &\text{Subtour 1: } 1 \to 5 \to 1, \\ &\text{Subtour 2: } 2 \to 4 \to 3 \to 2; \end{aligned}$$

see Figure 7.10(a).

The total tour length equals to $\max\{2,3\} + \max\{4,5\} + \max\{6,5\} + \max\{8,6\} + \max\{9,7\} = 31$. The algorithm then performs subtour patching with the minimum cost; see Appendix B or Pinedo [132] for details. The resulting tour is $1 \to 5 \to 3 \to 2 \to 4 \to 1$, so $\rho^* = (1, 5, 3, 2, 4)$;

k	1	2	3	4	5
$\overline{b}_k$	3	5	5	6	7
$\overline{a}_{\rho(k)}$	2	4	6	8	9
$\rho(k)$	5	4	2	3	1

Table 7.10. A Matching of Parameters $\overline{a}_k$ and $\overline{b}_k$.



(*a*) Two subtours (*b*) Subtours are patched

Figure 7.10. Subtour Patching.

see Figure 7.10(b). Note that the tour length is $\max\{9,6\}+\max\{2,3\}+\max\{8,7\}+\max\{6,5\}+\max\{4,5\} = 31$, i.e., a valid tour is formed without any additional cost. We now calculate the cycle time for the part schedule ρ^*. We set $\sigma = \rho^* = (1,5,3,2,4)$. Algorithm RobotWait in Section 7.2 can be used to solve the system of equations given in (7.44). A solution is $v_1 = v_4 = 1$, $v_2 = v_3 = v_5 = 0$. Using (7.45), we obtain

$$T^{\sigma}_{3,10} = Z_{3,10} + \sum_{i-1}^{5} \max\{\overline{a}_{\sigma(i+2)} - v_i, \overline{b}_{\sigma(i+1)}\} = 115 + 29 = 144.$$

The theorem below states a guarantee on the worst-case performance of algorithm Hard-CRM.

THEOREM 7.8 *For $RF_2^2|(free,A,MP,\ S_{q,r} \in \widetilde{\mathcal{S}}\ )|\mu$, let $\rho^* = (\rho^*(1), \rho^*(2), \ldots, \rho^*(n))$ be the schedule of parts $P_i \in \mathcal{P}$, found by algorithm Hard-CRM. Then the cycle time $T^{\rho^*}_{q,r}$ for MPS production for the schedule ρ^**

satisfies

$$T_{q,r}^{\rho^*} \leq \frac{3}{2} T_{q,r}^*, \tag{7.77}$$

where $T_{q,r}^*$ *is the optimal cycle time. The bound* $\frac{3}{2}T_{q,r}^*$ *is tight.*

Proof. Clearly, the theorem holds for $n = 2$. Thus, consider $n \geq 3$. Let $\sigma^* = (\sigma^*(1), \sigma^*(2), \ldots, \sigma^*(n))$ be an optimal solution to $RF_2^2|(free,A,MP, S_{q,r} \in \widetilde{\mathcal{S}})|\mu$, i.e.,

$$T_{q,r}^* = T_{q,r}^{\sigma^*}. \tag{7.78}$$

By Theorem 7.7, there always exists a schedule ρ such that

$$\sum_{i=1}^{n} \max\left\{\overline{a}_{\rho(i+1)}, \overline{b}_{\rho(i)}\right\} \leq \frac{3}{2}\left(T_{q,r}^{\sigma^*} - Z_{q,r}\right). \tag{7.79}$$

Furthermore, with ρ^* being a schedule which minimizes $\sum_{i=1}^{n} \max\left\{\overline{a}_{\rho(i+1)}, \overline{b}_{\rho(i)}\right\}$, we have

$$\sum_{i=1}^{n} \max\left\{\overline{a}_{\rho^*(i+1)}, \overline{b}_{\rho^*(i)}\right\} \leq \sum_{i=1}^{n} \max\left\{\overline{a}_{\rho(i+1)}, \overline{b}_{\rho(i)}\right\} \tag{7.80}$$

for any part schedule ρ. Hence, by (7.78)–(7.80), we have

$$\sum_{i=1}^{n} \max\left\{\overline{a}_{\rho^*(i+1)}, \overline{b}_{\rho^*(i)}\right\} \leq \frac{3}{2}\left(T_{q,r}^* - Z_{q,r}\right). \tag{7.81}$$

From (7.45), we have the following bound on the value of $T_{q,r}^{\rho^*}$, subject to a given schedule ρ^* of parts:

$$T_{q,r}^{\rho^*} \leq Z_{q,r} + \sum_{i=1}^{n} \max\{\overline{a}_{\rho^*(i+1)}, \overline{b}_{\rho^*(i)}\}. \tag{7.82}$$

Combining (7.81) and (7.82) immediately yields the desired bound (7.77).

To see that the bound is tight, consider the following instance of $RF_2^2|(free,A,MP, S_{q,r} \in \widetilde{\mathcal{S}})|\mu$. Let $\epsilon = \delta = \theta = 0$. There are four parts $P_i \in \mathcal{P}, i = 1, 2, 3, 4$, with the following values of a_i and b_i: $a_1 = b_1 = M$; $a_2 = b_2 = M$; $a_3 = b_3 = 0$; $a_4 = b_4 = 0$. As $\epsilon = \delta = \theta = 0$, we have $Z_{q,r} = \gamma_{q,r}^a = \gamma_{q,r}^b = 0$, as well as $\overline{a}_i = a_i$ and $\overline{b}_i = b_i$, $i = 1, 2, 3, 4$. A schedule which minimizes $\sum_{i=1}^{4} \max\left\{\overline{a}_{\rho(i+1)}, \overline{b}_{\rho(i)}\right\}$ is $\rho^* = (1, 2, 3, 4)$. For this schedule $T_{q,r}^{\rho^*} = 3M$ (a solution for the corresponding LP problem (7.47)–(7.50) is given by $u_1 = u_2 = u_3 = M$ and $u_4 = 0$,

$v_i = 0$, $i = 1, 2, 3, 4$). On the other hand, the optimum here is delivered by the schedule $\rho_{OPT} = (1, 3, 2, 4)$ with $T_{q,r}^{\rho_{OPT}} = 2M$ (a solution for the corresponding LP problem (7.47)–(7.50) is given by $v_1 = v_2 = M$ and $v_3 = v_4 = 0$, $u_i = 0$, $i = 1, 2, 3, 4$). Hence, the bound is $3M/2M = 3/2$. We remark that this example can be easily transformed to one with nonzero numeric data by setting $a_3 = b_3 = a_4 = b_4 = \alpha$, with α satisfying $M >> \alpha$, and by assigning sufficiently small values to ϵ, δ, and θ such that $Z_{q,r}$ is small relative to the corresponding processing times. The bound then approaches $3/2$ as M goes to infinity. ∎

REMARK 7.2 Note that the Gilmore-Gomory algorithm finds an optimal solution to $F_2|$*no-wait*$|C_t$ with no missing operations, i.e., the zero processing times are treated as positive but negligibly small values so that the processing route remains the same for all jobs. For $RF_2^2|$(*free,A,MP,*$S_{u,v} \in \mathcal{S}_0)|\mu$, the robot move sequence (CRM sequence) is generated by a 1-unit robot move sequence, which is executed repeatedly and identically. The identical execution of robot move operations for all parts directly implies the same processing route for all parts, and, therefore, the adequacy of the no-missing-operations assumption as implied by the Gilmore-Gomory algorithm.

We further observe that algorithm Hard-CRM suggests a heuristic procedure to solve $RF_2^2|$(*free,A,MP,*$S_{u,v} \in \mathcal{S}_0)|\mu$. From the results in Section 7.2 concerning our search for an optimal solution for the latter problem, we may restrict our attention to the MPS cycles based on 13 CRM sequences. For 7 out of these 13 cycles, the corresponding part scheduling problems admit an efficient solution, whereas each of the remaining six cycles generates a strongly NP-hard part-scheduling problem that can be solved approximately by algorithm Hard-CRM. By solving each of the polynomially solvable problems optimally and each of the NP-hard ones approximately, and then choosing the best solution among the 13 obtained, we find a solution to the general $RF_2^2|$(*free,A,MP,*$S_{u,v} \in \mathcal{S}_0)|\mu$ problem that is at most $3/2$ times the optimal value. Thus, Theorem 7.8 implies

COROLLARY 7.1 *For* $RF_2^2|$(*free,A,MP,*$S_{u,v} \in \mathcal{S}_0)|\mu$, *an* $O(n \log n)$ *algorithm based on Hard-CRM provides a worst-case bound of* $3/2$.

7.5 A Heuristic for Two-Machine Cells

We propose the following heuristic to solve $RF_2^2|(free,A,MP,S_{u,v} \in \mathcal{S}_0)|\mu$ over all CRM sequences.

Heuristic TwoCell

Input: An instance of $RF_2^2|(free,A,MP,S_{u,v} \in \mathcal{S}_0)|\mu$.

Output: A schedule $\sigma_{u,v}$ of parts $P_k \in \mathcal{P}$, the CRM sequence $S_{u,v}$, and the cycle time $T_{u,v}^{\sigma_{u,v}}$ of MPS processing in schedule $\sigma_{u,v}$ under the CRM sequence $S_{u,v}$.

Step 1: Form an arbitrary schedule σ of parts. Calculate $C_1 = 6n\epsilon + 3n\delta + \sum_{i=1}^{n}(a_i + b_i)$: the cycle time for $RF_2^2|(free,A,MP,S_{1,13})|\mu$.

Step 2: Let $\mathcal{S}' = \{S_{1,1}, S_{1,4}, S_{1,14}, S_{1,15}, S_{3,3}, S_{3,11}\}$. Use the Gilmore-Gomory algorithm to solve $RF_2^2|(free,A,MP,S_{u,v})|\mu$ for each CRM sequence $S_{u,v} \in \mathcal{S}'$. Denote by $\sigma_{u,v}$ the obtained part schedule corresponding to $S_{u,v}$. Find $C_2 = \min_{S_{u,v} \in \mathcal{S}'} T_{u,v}^{\sigma_{u,v}}$.

Step 3: Use algorithm Hard-CRM to solve $RF_2^2|(free,A,MP,S_{u,v})|\mu$ for each $S_{u,v} \in \widetilde{\mathcal{S}}$. Denote by $\sigma_{u,v}$ the obtained part schedule corresponding to $S_{u,v} \in \widetilde{\mathcal{S}}$. Find $C_3 = \min_{S_{u,v} \in \widetilde{\mathcal{S}}} T_{u,v}^{\sigma_{u,v}}$.

Step 4: Find $C_h = \min\{C_1, C_2, C_3\}$. Output C_h and the corresponding $S_{u,v}$ and $\sigma_{u,v}$. Stop.

The heuristic was tested on randomly generated problems. Our choice of data to test the performance of the heuristic satisfies condition (7.26). We generated random problem data under types I_1, I_2, I_3, and I_4, as follows:

I_1: The processing times are chosen randomly from $U[1, 5]$.
I_2: The processing times are chosen randomly from $U[1, 15]$.
I_3: The processing times are chosen randomly from $U[1, 25]$.
I_4: The processing times are chosen randomly from $U[1, 50]$.

Furthermore, for all problem instances, we set $\epsilon = 1.0, \delta = 3.0$, and $\theta = 0.25$. Because of the difficulty in finding optimal solutions to the problem, we tested the performance of TwoCell against a lower bound on the optimal cycle time computed as follows. (Note that the validity of this bound is governed by condition (7.26)).

Procedure LBCT$(a_i, b_i, i = 1, \ldots, n, \delta, , \epsilon, \theta)$
Step 1: $LB_1 = \sum_{i=1}^{n} \max\{3\delta + 6\epsilon + 2\theta, b_i + \theta + 2\epsilon\}$.
Step 2: $LB_2 = \sum_{i=1}^{n} \max\{3\delta + 6\epsilon + 2\theta, a_i + \theta + 2\epsilon\}$.
Step 3: $LB = \max\{LB_1, LB_2\}$. Stop.

Computer runs, each consisting of five problem instances, were carried out. Let f_k be the cycle time obtained by TwoCell for the kth problem in a run, and let f_k^* be the lower bound from LBCT for the kth problem. Let

$$e_k = \frac{f_k - f_k^*}{f_k^*} 100, \quad e = \sum_{k=1}^{5} \frac{e_k}{5},$$
$$\ell = \min_{1 \le k \le 5} \{e_k\}, \quad \text{and} \quad u = \max_{1 \le k \le 5} \{e_k\}.$$

Each row in Table 7.11 corresponds to the results of five instances. The third, fourth, and fifth columns show the minimum, the mean, and the maximum relative percentage error, respectively, for the heuristic. On average, TwoCell performs well with respect to the lower bound. We note that the mean relative errors are all less than 10%. The smallest and the largest relative errors observed were 0.0% and 14.69%, respectively, over all the 80 instances reported in Table 7.11.

7.6 Comparison of Productivity: Single-Gripper Vs. Dual-Gripper Cells

In this section, we consider algorithms to solve the problem of scheduling an MPS in two-machine robotic cells with single-gripper and with dual-gripper robots. This allows us to assess the improvement in productivity effected by the use of a dual gripper over a single gripper. Hall et al. [75] solve the two-machine robotic cell problem with single gripper: $RF_2^1|(free, A, MP, cyclic\text{-}n)|\mu$. They show that the problem here is one of deciding on the parts to process under the sequences $S_{1,13}$ and $S_{1,18}$, which are the only possible sequences in the single-gripper case. They devise an $O(n^4)$ algorithm, MinCycle, which solves the problem optimally. Aneja and Kamoun [7] modify their algorithm and reduce the time complexity to $O(n \log n)$. Algorithm MinCycle is described in Chapter 6.

Algorithms TwoCell and MinCycle were tested on randomly generated problems under the instance types I_1, I_2, I_3, and I_4, described above.

n	Instance	ℓ	e	u
10	I_1	0.0	0.0	0.0
15	I_1	0.0	0.0	0.0
25	I_1	0.0	0.0	0.0
50	I_1	0.0	0.0	0.0
10	I_2	0.00	0.57	0.96
15	I_2	0.00	0.06	0.32
25	I_2	0.00	0.37	0.70
50	I_2	0.32	0.54	0.77
10	I_3	0.75	3.17	6.53
15	I_3	0.00	3.93	6.79
25	I_3	3.60	4.75	6.61
50	I_3	3.04	5.82	8.01
10	I_4	0.65	5.72	13.10
15	I_4	2.24	7.11	10.46
25	I_4	1.99	9.83	14.69
50	I_4	5.64	10.01	13.16

Table 7.11. Performance Evaluation of Heuristic TwoCell.

Computer runs, each consisting of five instances, were carried out. Let f_k^d be the dual-gripper cycle time obtained by heuristic TwoCell for the kth problem in a run, and let f_k^s be the single-gripper cycle time obtained by algorithm MinCycle for the kth problem. Let

$$\begin{aligned} h_k &= \frac{f_k^s - f_k^d}{f_k^s} 100, \quad h = \sum_{k=1}^{5} \frac{h_k}{5} \\ \ell_h &= \min_{1 \le k \le 5} \{h_k\}, \text{and} \quad u_h = \max_{1 \le k \le 5} \{h_k\}. \end{aligned}$$

Table 7.12 compares the cycle times obtained for single-gripper and dual-gripper cells. Each row in Table 7.12 corresponds to the results of five instances. The third, fourth, and fifth columns show the minimum, the mean, and the maximum relative percentage improvements, respectively, for the dual-gripper cell with respect to the single-gripper cell. The results of computational experiments indicate that, on average, a dual-gripper cell provides a significant productivity improvement over a single-gripper cell. Note that the mean relative improvements are all above 18%. The smallest and the largest relative improvements observed

were 14.25% and 36.64%, respectively, over all 80 problems reported in Table 7.12.

n	Instance	ℓ_h	h	u_h
10	I_1	20.10	23.76	26.89
15	I_1	23.01	23.32	23.77
25	I_1	21.72	24.11	26.47
50	I_1	23.12	23.87	24.24
10	I_2	33.23	34.37	36.64
15	I_2	31.96	33.56	34.75
25	I_2	30.89	32.48	34.11
50	I_2	32.77	33.43	33.77
10	I_3	26.08	31.10	35.61
15	I_3	26.54	30.63	32.38
25	I_3	28.96	30.06	30.77
50	I_3	28.51	29.44	31.08
10	I_4	17.59	21.75	25.18
15	I_4	18.97	21.04	24.49
25	I_4	14.25	18.57	24.43
50	I_4	14.83	17.55	21.10

Table 7.12. Productivity Comparison of Dual Vs. Single Gripper Robot Cells.

7.7 An Extension to m-Machine Robotic Cells

In this section we address problem $RF_m^2|(free,A,MP,cyclic\text{-}n)|\mu$ for $m \geq 2$. Drobouchevitch et al. [50] show that for single-part-type production in a dual-gripper cell, the number of 1-unit cycles increases with the number of machines at a much faster rate than that for a single-gripper cell. Consequently, as m increases, the dual-gripper problem is, in general, much more complex than that for a single-gripper. Furthermore, in multiple-part-type production, the scheduling of parts adds to the complexity.

For $RF_m^2|(free,A,MP,cyclic\text{-}n)|\mu$, let $p_{r,j}$ denote the processing time of part P_r on machine M_j. Of special interest is the variant of the problem with small gripper switching time, i.e.,

$$\theta \leq \min\{\delta, \min_{r,j} p_{r,j}\}. \tag{7.83}$$

Drobouchevitch et al. [50] proved that an m-machine analogue of the 1-unit cycle $C_{3,10}$, referred to as C_m^d, is optimal for single-part-type production in an m-machine dual-gripper cell $(RF_m^2|(free,A,cyclic\text{-}1)|\mu)$ under condition (7.83). The cycle C_m^d is a straightforward extension of cycle $C_{3,10}$ to the m-machine case, and can be described as follows (See Chapter 4):

Cycle C_m^d

Step 1: Pick up a part P_{i+m} from Input device at I/O.

Step 2: For k from 1 to m:

Go to machine M_k. If necessary, wait for a part P_{i+m-k} on M_k to complete processing.

Unload a part from M_k.

Switch gripper.

Load a part onto M_k.

Step 3: Go to I/O; drop a part P_i onto Output buffer.

The purpose here is to solve $RF_m^2|(free,A,MP,S \in \mathcal{S}_0^m)|\mu$ in the class of all CRM sequences $S \in \mathcal{S}_0^m$, where $\mathcal{S}_0^m$ consists of all CRM sequences associated with 1-unit cycles for m-machine cells.

In Section 7.3 we showed that for $RF_m^2|(free,A,MP,S_{u,v} \in \mathcal{S}_0)|\mu$ under condition (7.83), the CRM sequence $S_{3,10}$ provides an optimal solution when $m = 2$. We also showed (in Section 4.4) that C_m^d is optimal for single-part-type cells $(RF_m^2|(free,A,cyclic\text{-}1)|\mu)$ under condition (7.83). These results have not been extended for m-machine cells with multiple part-types. However, the results from Sections 7.3 and 7.4.1 suggest that for $RF_m^2|(free,A,MP,S \in \mathcal{S}_0^m)|\mu$ under condition (7.83), the CRM sequence (denoted as S_m^d) obtained by concatenating n identical C_m^d sequences is a promising candidate to consider. Lemma 7.7 provides the cycle time expression for the CRM sequence S_m^d.

LEMMA 7.7 *For $RF_m^2|(free,A,MP,S_m^d)|\mu$, the cycle time of an MPS cycle in a schedule σ is $T_{3,10}^{\sigma} = \sum_{i=1}^{n} T_{3,10}^{m\sigma(i)\sigma(i+1)}$, where*

$$T_{3,10}^{m\sigma(i)\sigma(i+1)} = (m+1)\delta + 2(m+1)\epsilon + m\theta + \sum_{k=1}^{m} w_k^{i+m+1-k}, \quad (7.84)$$

and for $j = 1, \ldots, m$,

$$w_j^{i-j+1} = \max\left\{0, p_{\sigma(i+1-j),j} - (m+1)\delta - 2m\epsilon - (m-1)\theta - \sum_{k=j+1}^{m} w_k^{i-k} - \sum_{k=1}^{j-1} w_k^{i-k+1}\right\}. \quad (7.85)$$

Since the part scheduling problem for $RF_m^2|(free,A,MP,S_m^d)|\mu$ is NP-complete in the strong sense for $m = 2$, research has focused on the design of good heuristics. It is an attractive research goal to investigate the possibility of extending the results from Sections 7.4 and 7.5 to the case of an m-machine cell. One interesting option to consider is the application of the proposed heuristic procedure for $RF_2^2|(free,A,MP,S_{3,10})|\mu$ to solve an m-machine instance by relaxing the latter to obtain an artificial two-machine problem. Our heuristic, referred to as MCell, for m-machine cells is as follows: Given an instance of $RF_m^2|(free,A,MP,S_m^d)|\mu$, we form $(m-1)$ artificial $RF_2^2|(free,A,MP,S_{3,10})|\mu$ instances by setting

$$p_{i,j}^{new} = 0, \quad j = 1, 2, \ldots k-1, k+2, k+3, \ldots m, \quad i \in \mathcal{P};$$
$$p_{i,j}^{new} = p_{i,j}, \quad j = k, k+1, \quad i \in \mathcal{P},$$

and varying k from 1 to $m-1$. The new processing times $p_{i,j}^{new}$ are then incorporated into the expressions (7.84)–(7.85). As a result, we have $(m-1)$ problems, all equivalent to $RF_2^2|(free,A,MP,S_{3,10})|\mu$, that are solved approximately by algorithm Hard-CRM of Section 7.4 (with some proper tuning of values $\overline{a}_k = \max\{0, a_k - \gamma_{u,v}^a\}$ and $\overline{b}_k = \max\{0, b_k - \gamma_{u,v}^b\}$ in the function $\sum \max\{\overline{a}_{\sigma(i+1)}, \overline{b}_{\sigma(i)}\}$ to be minimized by the Gilmore-Gomory algorithm, subject to problem data). Note that the condition of preserving the original processing times of the parts on two successive machines is required to adequately relax the m-machine problem to the corresponding two-machine problem under the CRM sequence $S_{3,10}$. By solving each of $(m-1)$ artificial problems approximately, we obtain $(m-1)$ part schedules. We then calculate the cycle time for each of these schedules for $RF_m^2|(free,A,MP,S_m^d)|\mu$. The part schedule that delivers the smallest cycle time determines the final solution.

Heuristic MCell was tested on randomly generated problems under instance types I_1, I_2, I_3, and I_4, as described in Section 7.5. In particular,

condition (7.83) is valid for all instances generated for the computational experiments. The performance of the heuristic was tested against a lower bound on the optimal cycle time obtained as follows.

LEMMA 7.8 *For $RF_m^2|(free,A,MP,S \in \mathcal{S}_0^m)|\mu$ under condition (7.83), the cycle time $T(S)$ of any CRM sequence S has a lower bound LB_m:*

$$T(S) \geq LB_m = \max_{1\leq j\leq m}\left\{\sum_{r=1}^{n}\max\{p_{r,j}+\theta+2\epsilon, (m+1)\delta+2(m+1)\epsilon+m\theta\}\right\}. \quad (7.86)$$

Proof. Let S be the CRM sequence constructed from an arbitrary 1-unit robot move cycle C. Let M_k be an arbitrary machine. We prove the lemma by demonstrating that

$$T(S) \geq \sum_{r=1}^{n}\max\left\{p_{r,k}+\theta+2\epsilon,\ (m+1)\delta+2(m+1)\epsilon+m\theta\right\}. \quad (7.87)$$

Let $\sigma = (\sigma(1), \sigma(2), ..., \sigma(n))$ be a schedule of parts, and let us consider a complete execution of CRM sequence S. Let t_1 and t_2, $t_1 < t_2$, be any two successive instants in time when machine M_k starts processing of a part $\sigma(1)$, so that $t_1 + T(S) = t_2$. Note that the CRM sequence S is a concatenation of n copies of C. We denote by C_i the ith copy of C, whose execution starts at the moment just after part $\sigma(i)$ is loaded onto machine M_k and ends at the moment when the next part $\sigma(i+1)$ in the schedule is just loaded onto the same machine. We denote by $T(C_i)$ the time taken by a complete execution of C_i. Thus, $T(S) = \sum_{i=1}^{n} T(C_i)$. To prove (7.87), it suffices to show that

$$T(C_i) \geq \max\left\{p_{\sigma(i),k}+\theta+2\epsilon,\ (m+1)\delta+2(m+1)\epsilon+m\theta\right\}.$$

Consider an execution of C_i. Recall that M_j^- and M_j^+, $0 \leq j \leq m$, denote operations "load machine M_j" and "unload machine M_j", respectively. Let $O^- = \cup_{0\leq j\leq m} M_j^-$ and $O^+ = \cup_{0\leq j\leq m} M_j^+$. The robot move sequence for C_i can then be defined by a sequence $\omega = \left(O_1, O_2, ..., O_{2(m+1)}\right)$, where $O_j \in O^- \cup O^+$, $O_{j_1} \neq O_{j_2}$ for any $j_1 \neq j_2$, and $O_{2(m+1)} = M_k^-$.

For C_i, let τ_i be the time taken by a partial execution of C_i from the moment when the robot is just starting to unload machine M_k (operation

M_k^+) until the moment when the robot just finishes loading machine M_k (operation M_k^-) . Obviously, $T(C_i) \geq \tau_i + p_{\sigma(i),k}$. Furthermore, if M_k^- immediately follows M_k^+ in the ω operation, then $\tau_i = 2\epsilon + \theta$. Otherwise, there is at least one operation executed after operation M_k^+ and before operation M_k^-. Since such an operation is executed on another machine rather than on M_k, it requires the robot to travel from M_k to that machine. Hence, $\tau_i \geq 2\epsilon + \delta \geq 2\epsilon + \theta$ by (7.83). Therefore, $T(C_i) \geq p_{\sigma(i),k} + \theta + 2\epsilon$.

Thus, what is left to show is that

$$T(C_i) \geq (m+1)\delta + 2(m+1)\epsilon + m\theta. \tag{7.88}$$

Given the sequence $\omega = \left(O_1, O_2, ..., O_{2(m+1)}\right)$ associated with C_i, by a *contribution* $\Delta(O_j)$ of a particular operation O_j to $T(C_i)$, we mean the time taken by a partial execution of C_i from the moment when the robot has just completed operation O_{j-1} to the moment when it is about to start the execution of operation O_j. We then have

$$T(C_i) = 2(m+1)\epsilon + \sum_{j=1}^{2(m+1)} \Delta(O_j). \tag{7.89}$$

We first observe that $\Delta(M_0^-) + \Delta(M_0^+) \geq \delta$, since the execution of at least one of operations M_0^- and M_0^+ necessarily contributes (to $T(C_i)$) the robot travel time between some machine and I/O. Consider now operations $O_j \in \{O^- \cup O^+\} \setminus \left\{M_0^-, M_0^+\right\}$. If two consecutive operations O_{j-1} and O_j are executed on the same machine M_ℓ, then the contribution $\Delta(O_j)$ is equal to θ if $O_{j-1} = M_\ell^+$ and $O_j = M_\ell^-$, or to part processing time $p_{r,\ell} \geq \theta$ (by (7.83)) if $O_{j-1} = M_\ell^-$ and $O_j = M_\ell^+$. Clearly, the total number γ of such operations O_j for which O_{j-1} is executed on the same machine is at most m. On the other hand, if two consecutive operations O_{j-1} and O_j are executed on different machines, then the contribution $\Delta(O_j)$ includes the robot travel time between the machines on which these operations are executed, so it is at least δ. The total number of such operations O_j for which O_{j-1} is executed on a different machine is equal to $2m - \gamma \geq m$. By (7.83), $(m-\gamma)\,\delta \geq (m-\gamma)\,\theta$, and so we have

$$\sum_{j=1}^{2(m+1)} \Delta(O_j) \geq \delta + \gamma\theta + (2m - \gamma)\,\delta \geq m\theta + (m+1)\,\delta.$$

Combining this relation with (7.89) yields (7.88). ∎

As before, computer runs, each consisting of 10 instances, were carried out on randomly generated data under types I_1, I_2, I_3, and I_4. The performance of heuristic MCell was compared with the lower bound LB_m established in Lemma 7.8. Each row in Table 7.13 corresponds to a summary of the results for 10 instances. Table 7.13 has the same

n	I_i	$m = 3$			$m = 5$			$m = 8$		
		ℓ	e	u	ℓ	e	u	ℓ	e	u
10	I_3	0.00	2.42	4.48	0.00	0.00	0.00	0.00	0.00	0.00
15	I_3	0.45	2.35	4.56	0.00	0.00	0.00	0.00	0.00	0.00
25	I_3	0.44	2.90	4.22	0.00	0.00	0.00	0.00	0.00	0.00
50	I_3	2.84	3.59	4.33	0.00	0.00	0.00	0.00	0.00	0.00
10	I_4	0.51	7.46	17.45	1.03	5.15	14.66	0.26	0.90	1.83
15	I_4	2.24	10.65	16.83	2.72	7.09	12.69	0.59	1.33	2.37
25	I_4	5.48	10.89	17.00	7.06	10.57	13.52	0.86	1.35	2.00
50	I_4	10.35	13.73	16.16	8.85	12.33	16.26	1.15	1.78	2.25

Table 7.13. Performance Evaluation of Heuristic MCell.

format as Table 7.11. For example, the third, fourth, and fifth columns show the minimum, the mean and the maximum relative percentage error, respectively, for the three-machine case ($m = 3$). On average, heuristic MCell performed well with respect to the lower bound. Note that the mean relative errors are all less than 14%. The smallest and the largest relative errors observed were 0% and 17.45%, respectively, over all the 80 problems reported in Table 7.13. The relative errors are 0% for all the instances tested under types I_1 and I_2, so they are not reported in Table 7.13. In general, relative errors are small for instances with smaller part processing times (instances I_1 and I_2). The relative error grows as the part processing times increase (I_3 and I_4). For instances under I_1 and I_2, the term of LB_m that gives the total robot activity time ($(m+1)\delta + 2(m+1)\epsilon + m\theta$) dominates the processing time term ($p_{r,j} + \theta + 2\epsilon$). Consequently, under I_1 and I_2 the optimal

solutions are closer to LB_m. On the other hand, under I_3 and I_4 the part processing time term dominates the total robot activity time. Moreover, the optimal solution depends on the robot move sequence as well as the part schedule. Given that the lower bound LB_m does not take into account part scheduling, the error estimates under I_4 in Table 7.13 may not be as large as they appear to be. Thus, the performance of MCell might be better than that indicated in Table 7.13. Note that as the number of machines increases, the relative error decreases; again, the term $(m+1)\delta + 2(m+1)\epsilon + m\theta$ may dominate in LB_m.

Chapter 8

MULTIPLE-ROBOT CELLS

Two early studies of systems with multiple robots focus on cells that are not flowshops: the robots move to different workstations to perform a variety of assembly functions for different parts by using different tools; conflicts arise over the tools used and the paths traveled. Maimon and Nof [118] discuss enumerative algorithms that determine how high-level decisions such as work allocation, cooperation, and the robots' routes of travel are made by the control computer during operations. Nof and Hannah [126] analyze how productivity is affected by different types of cooperation among the robots and by the level of resource sharing by the robots.

In Chapter 9, we shall discuss how Kats and Levner [97] address the multiple robot scheduling problem in no-wait cells. A previous work by the same authors [95] proposes an algorithm to find the minimum number of robots needed to meet a given throughput. However, the algorithm allows for different robots to move along the same path, i.e., to travel the same cycle. This scheme may be suitable for a system with automated guided vehicles or with computer-controlled hoists, but in most robotic cells there is insufficient space for such movements by the robots.

This chapter summarizes the work of Geismar et al. [64] that addresses the problem of scheduling a robotic cell with parallel machines and multiple robots working concurrently in a free-pickup cell producing a single part-type. In terms of the classification of Chapter 2, the

problem addressed here is $RF_{m,r}|(free,E,cyclic\text{-}k)|\mu$. We begin with the physical description of cells with multiple robots (Section 8.1). An extension of LCM cycles (Chapter 5) is analyzed (Sections 8.2 and 8.3), and then compared to a heuristic dispatching method (Section 8.4) used by a Dallas-area semiconductor equipment manufacturer by using software simulation and simulated data for cells with Euclidean travel-time (Sections 8.5 and 8.6).

8.1 Physical Description of a Multiple-Robot Cell

The r robots in the cell are denoted $R_1,\ R_2,\ldots,\ R_r$. Before processing begins, each robot is assigned stages to load and to unload. Let $\mathcal{R}_j$ be the set of indices of stages unloaded by robot $R_j,\ j = 1,\ldots,r$; i.e., R_j, and only R_j, performs activities $A_i,\ i \in \mathcal{R}_j$. There are specifically designated *shared* stages whose machines perform a processing function just like any other stage, but each shared stage is loaded by one robot and unloaded by another. The set of indices for these shared stages is denoted Q.

For example, a Dallas-area semiconductor equipment manufacturer has produced a cell that has 18 stages, some of which have parallel machines. The machines are housed in modules that resemble racks used for computer hardware. Each module contains the machines of one stage. The machines of a particular stage are stacked atop one another (see Figure 8.1(a)). This simplifies programming a robot's movements because the horizontal (x, y) coordinates of the machines of a particular stage are all the same. Only the vertical (z) coordinate changes for different machines of the same stage. Some modules contain only one machine ($m_i = 1$), but the one-stage-per-module design allows machines to be added to stages with minimal disruption to the layout or to the programming of the robots. These modules are arranged in a horseshoe configuration (see Figure 8.1(b)).

This cell also contains three robots (R_1, R_2, R_3) that transfer the silicon wafers between the various stages. Each robot is assigned specific stages whose machines it loads or unloads. These assignments are determined by the cell's configuration: its long, narrow shape limits the modules (hence the stages) that each robot can reach. Therefore, the

robots' assignments are fixed input parameters to the robot scheduling problem.

A wafer begins processing when robot R_1 unloads it from the input device and loads it onto some machine in stage 1. After stage 1 processing, R_1 unloads the wafer and loads it onto a machine in stage 2. Robot R_2 unloads it (stage 2 is a shared stage) and successively loads it onto machines in stages 3, 4, and 5. Robot R_3 unloads the wafer (stage 5 is shared) and successively loads it onto stages 6, 7, 8, 9, 10, and 11. R_2 unloads stage 11 (stage 11 is shared) and transfers the wafer through stages 12, 13, and 14, before loading it onto some machine of the shared stage 15. Robot R_1 unloads stage 15, carries the wafer through stages 16, 17, and 18, and then places the completed wafer into the output device. Hence, for this cell, $\mathcal{R}_1 = \{0, 1, 15, 16, 17, 18\}$, $\mathcal{R}_2 = \{2, 3, 4, 11, 12, 13, 14\}$, $\mathcal{R}_3 = \{5, 6, 7, 8, 9, 10\}$, and $Q = \{2, 5, 11, 15\}$.

The total floor space covered by a robot's movements is called its *envelope*. The cell is designed so that the envelopes of the robots intersect only in small areas around each of the modules containing the shared stages. To ensure that two robots do not collide in one of these four areas, a time buffer β is specified. If robot R_k is loading or unloading a machine in stage i, $i \in Q$, then robot R_j, $j \neq k$, cannot move to any machine in stage i until R_k has completed its operation and β time units have passed.

This non-collision policy is implemented by using a *token* τ_i for each shared stage $i \in Q$. Before moving to a machine in a shared stage, robot R_j must first allocate the stage's token. If the token is currently

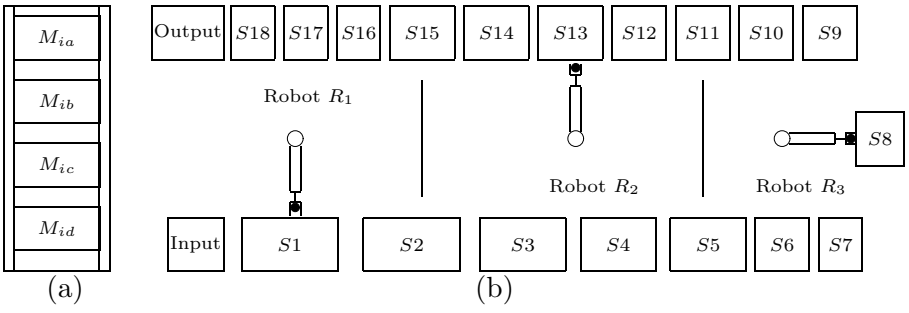


Figure 8.1. (a) Module Containing the Machines of a Stage. (b) Three-Robot Cell with 18 Stages.

held by another robot R_k, then R_j waits at its current location until the token is released by R_k. Token τ_i is released by R_k β time units after the completion of its operation (loading or unloading) at a machine in stage i. When loading a machine in stage i, R_j's sequence of actions is (1) unload a machine in stage $i-1$, (2) allocate token τ_i, (3) move to a machine in stage i, and (4) load that machine in stage i. The actions of releasing a currently held token or of allocating a free token are assumed to require zero time, since they are computer instructions.

8.2 Cycles in Multiple-Robot Cells

Unlike a single-robot cell, the operations of an r-robot cell cannot be accurately represented by a single cycle of robot moves. Rather, each robot has its own repeated sequence of moves that it executes concurrently with those of the other robots. In an r-robot cell, a cycle is defined to be the combination of these r repeated sequences. For robot R_j, its repeated sequence will contain activities $A_{i\gamma\eta}$, where $i \in \mathcal{R}_j$, $\gamma \in \{a, \ldots, \alpha(m_i)\}$, $\eta \in \{a, \ldots, \alpha(m_{i+1})\}$. As an example, the definition of π_D (Chapter 3) for a multi-robot cell is as follows.

DEFINITION 8.1 In an r-robot cell with one machine in each stage, the cyclic solution (or cycle) π_D is the collection of the r repeated sequences $\pi_D^j = (A_{i_{j_1}}, A_{i_{j_2}}, \ldots, A_{i_{j_{|\mathcal{R}_j|}}})$, $j = 1, \ldots, r$; $\mathcal{R}_j = \{i_{j_1}, i_{j_2}, \ldots, i_{j_{|\mathcal{R}_j|}}\}$, and the sequence $i_{j_1}, i_{j_2}, \ldots, i_{j_{|\mathcal{R}_j|}}$ is decreasing. Furthermore, $\cup_{j=1}^{r} \mathcal{R}_j = \{0, 1, \ldots, m\}$ and $\mathcal{R}_j \cap \mathcal{R}_k = \emptyset$, $1 \le j \le k \le r$.

For the three-robot cell of Figure 8.1(b), if each stage has only one machine, then $\pi_D^1 = (A_{18}, A_{17}, A_{16}, A_{15}, A_1, A_0)$, $\pi_D^2 = (A_{14}, A_{13}, A_{12}, A_{11}, A_4, A_3, A_2)$, and $\pi_D^3 = (A_{10}, A_9, A_8, A_7, A_6, A_5)$.

To define the throughput of a cycle in the multi-robot cell, consider the three-robot cell of Figure 8.1(b). Observe that the rate at which R_1 places parts into the output device must equal the long-term average rate that R_1 unloads parts from stage 15. This value in turn equals the rates for R_2's loading of stage 15 and its unloading of stage 11, which, of course, equals the rate at which R_3 loads stage 11. Hence, if each robot's throughput is defined as the long-term average number of parts placed into its last stage (R_1's placement of parts into O, R_2's placement into stage 15, and R_3's into stage 11) per unit time, then the robots have equal throughput, which is then the cell's throughput.

Therefore, the synchronization of the robots is similar to the synchronization of tasks on an assembly line: cycle time is determined by the slowest worker. If robot R_j requires less time to complete its repeated sequence than does an adjacent robot R_k, then R_j will be forced to wait at one of the stages shared by these two robots. R_j will either wait for a machine in a shared stage to complete processing so that it can be unloaded, or R_j will wait at another shared stage for R_k to unload a machine so that R_j can reload it. It is these types of waiting that cause unbalanced robots to have equal throughput.

Cycle π_{LD} of Chapter 5 can be extended to an r-robot cell with $m_i \geq 1$ machines in stage i, $i \in M$. Like π_D, π_{LD} is a collection of r repeated sequences. Furthermore, in π_{LD} each robot visits stages in the same order as it does in π_D. In contrast to $\mathcal{R}_j$, let $\bar{\mathcal{R}}_j$ be the set of all stages whose machines are *loaded* by R_j, $j = 1, \ldots, r$. In π_{LD}, in each instance of its repeated sequence, the number of times that robot R_j unloads each stage $i \in \mathcal{R}_j$ and loads each stage $i \in \bar{\mathcal{R}}_j$ is λ_j, the least common multiple (LCM) of the number of machines in each of those stages either loaded or unloaded by R_j. Thus, $\lambda_j = LCM_{i \in \mathcal{R}_j \cup \bar{\mathcal{R}}_j}[m_i]$.

The repeated sequence for robot R_j in cycle π_{LD} can be viewed as λ_j blocks, each with $|\mathcal{R}_j|$ activities. In each of the blocks, the first index of the activities (indicating the unloaded stage) follow the decreasing sequence of π_D, i.e., $i_{j_1}, i_{j_2}, \ldots, i_{j_{|\mathcal{R}_j|}}$. The second and third indices (indicating the machines that are unloaded and loaded) are ordered so that each machine is loaded immediately after it is unloaded (activity $A_{i-1,\phi\gamma}$, for some $\phi \in \{a, \ldots, \alpha(m_{i-1})\}$, immediately follows $A_{i\gamma\eta}$ for $i \notin Q$, for some $\eta \in \{a, \ldots, \alpha(m_{i+1})\}$), and the machines of each stage are loaded cyclically and unloaded in the same order. This implies that for each stage $i \in \bar{\mathcal{R}}_j$, each of its machines is loaded λ_j/m_i times. In the example cell, if $m_5 = 4$, $m_6 = 3$, $m_7 = 2$, $m_8 = m_9 = m_{10} = 1$, and $m_{11} = 6$, then $\lambda_3 = 12$ and the repeated sequence for robot R_3 is

$$\begin{aligned}\pi^3_{LD} = (&A_{10*a}, A_{9**}, A_{8**}, A_{7a*}, A_{6aa}, A_{5aa},\\ &A_{10*b}, A_{9**}, A_{8**}, A_{7b*}, A_{6bb}, A_{5bb},\\ &A_{10*c}, A_{9**}, A_{8**}, A_{7a*}, A_{6ca}, A_{5cc},\\ &A_{10*d}, A_{9**}, A_{8**}, A_{7b*}, A_{6ab}, A_{5da},\\ &A_{10*e}, A_{9**}, A_{8**}, A_{7a*}, A_{6ba}, A_{5ab},\\ &A_{10*f}, A_{9**}, A_{8**}, A_{7b*}, A_{6cb}, A_{5bc},\\ &A_{10*a}, A_{9**}, A_{8**}, A_{7a*}, A_{6aa}, A_{5ca},\end{aligned}$$

$$A_{10*b}, A_{9**}, A_{8**}, A_{7b*}, A_{6bb}, A_{5db},$$
$$A_{10*c}, A_{9**}, A_{8**}, A_{7a*}, A_{6ca}, A_{5ac},$$
$$A_{10*d}, A_{9**}, A_{8**}, A_{7b*}, A_{6ab}, A_{5ba},$$
$$A_{10*e}, A_{9**}, A_{8**}, A_{7a*}, A_{6ba}, A_{5cb},$$
$$A_{10*f}, A_{9**}, A_{8**}, A_{7b*}, A_{6cb}, A_{5dc}).$$

π_{LD} is the collection of the r repeated sequences π_{LD}^j, $j = 1, \ldots, r$. In keeping with the definition of a cycle (Chapter 3), π_{LD} completes when the cell returns to its state at the beginning of π_{LD}. Hence, cycle π_{LD} produces $\lambda = LCM_{1 \le j \le r}[\lambda_j]$ parts, and in one complete π_{LD} cycle, π_{LD}^j is performed λ/λ_j times.

8.3 Cycle Times

The one-stage-per-module design implies that a robot's travel time between two machines $M_{i\gamma}$ and $M_{j\eta}$ is independent of the values of $\gamma \in \{a, \ldots, \alpha(m_i)\}$ and $\eta \in \{a, \ldots, \alpha(m_j)\}$: the time required for a vertical movement is small in comparison to that for horizontal movement. Hence, the only specification needed is that of the travel times δ_{ij} between stages i and j, $0 \le i, j \le m+1$. The only restrictions on these travel times are that they are symmetric ($\delta_{ij} = \delta_{ji}$) and satisfy the triangle inequality ($\delta_{ij} + \delta_{jk} \ge \delta_{ik}$). This leads to a lower bound for the per unit cycle time.

THEOREM 8.1 *In a robotic cell with parallel machines, r robots, and a fixed assignment of stages to robots, we have the following lower bound for the per unit cycle time:*

$$\frac{T(\pi)}{k} \ge \max \left\{ \max_{1 \le j \le r} \left\{ \sum_{i \in \mathcal{R}_j} (\delta_{i,i+1} + 2\epsilon) + \sum_{i \in \mathcal{R}_j \setminus Q} \min\{p_i, \min_{h \in \bar{\mathcal{R}}_j} \delta_{hi}\} + \sum_{i \in \mathcal{R}_j \cap (Q \cup \{0\})} \min_{h \in \bar{\mathcal{R}}_j} \delta_{hi} \right\}, \max_{i \notin Q} \left\{ \frac{p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon}{m_i} \right\}, \max_{i \in Q} \left\{ \frac{p_i + 2\epsilon + \beta + \delta_{i-1,i}}{m_i} \right\} \right\},$$

where k is the number of units produced by one iteration of cycle π.

Proof. Consider the first argument. For each part produced, each robot R_j performs an activity that unloads a machine in stage i, $\forall i \in \mathcal{R}_j$. Activity $A_{i\gamma\eta}$ requires time $\delta_{i,i+1} + 2\epsilon$: R_j unloads machine $M_{i\gamma}$ (time ϵ), travels to $M_{i+1,\eta}$ (time $\delta_{i,i+1}$), and loads $M_{i+1,\eta}$ (time ϵ). Before unloading a non-shared machine in stage i, a robot can wait for that machine's entire processing (time p_i) or move to that machine (at least $\min_{h \in \bar{\mathcal{R}}_j} \delta_{hi}$). Before unloading a shared machine or the input device, a robot moves to that machine or device.

The second argument is the minimum time required to complete a cycle if a robot waits at a non-shared machine. The time between two loadings of machine $M_{i\gamma}$ requires at minimum $M_{i\gamma}$'s processing (p_i), unloading the part (ϵ), traveling to some machine in stage $i+1$ ($\delta_{i,i+1}$), loading the part onto that machine (ϵ), traveling to some machine in stage $i-1$ ($\delta_{i+1,i-1}$), unloading a part from that machine (ϵ), traveling to $M_{i\gamma}$ ($\delta_{i-1,i}$), and loading the part onto $M_{i\gamma}$ (ϵ). In a k-unit cycle, the m_i machines in stage i are loaded a total of k times. Therefore, the minimum cycle time is achieved by loading each machine in stage i k/m_i times, which leads to the per unit cycle time of the second argument.

The minimum time between successive loadings of shared machine $M_{i\gamma}$ is $p_i + 2\epsilon + \beta + \delta_{i-1,i}$, obtained as follows. p_i time units after $M_{i\gamma}$ is loaded by robot R_j, robot R_k unloads the part (requiring time ϵ). β time units after unloading the part from $M_{i\gamma}$, R_k releases token τ_i. Hence, $p_i + \epsilon + \beta$ time units after its previous loading of $M_{i\gamma}$, robot R_j acquires τ_i . Once R_j has been allocated τ_i, R_j can transport the part it has unloaded from some machine in stage $i-1$ to $M_{i\gamma}$ ($\delta_{i-1,i}$) and load $M_{i\gamma}$ (ϵ). Dividing by m_i yields the required lower bound on the per unit cycle time. ∎

The LCM cycle π_{LD} reloads each machine as soon as possible after unloading it. It is, therefore, reasonable to expect this cycle to be optimal under certain conditions. We first derive the cycle time of π_{LD}. The lower bound on the per unit cycle time, established in Theorem 8.1, then helps us identify conditions under which π_{LD} is optimal. We provide two such conditions in Corollaries 8.1 and 8.2.

The expression for the per unit cycle time of π_{LD} has an argument that represents each robot's per unit cycle time if it never waits at a machine and arguments that represent each robot's per unit cycle time if one waits at either non-shared or shared machines. Recall that in π_{LD},

robot R_j, $j = 1, \ldots, r$, travels the sequence of stages $i_{j_1}, i_{j_2}, \ldots, i_{j_{\mathcal{R}_j}}$, where $i_{j_1} > i_{j_2} > \cdots > i_{j_{|\mathcal{R}_j|}}$.

THEOREM 8.2 *The per unit cycle time for π_{LD} is*

$$\frac{T(\pi_{LD})}{\lambda} = \max\left\{ \max_{1\le j\le r} \left\{ \sum_{k=1}^{|\mathcal{R}_j|} (\delta_{i_{j_k}, i_{j_k}+1} + \delta_{i_{j_k}+1, i_{j_{k+1}}} + 2\epsilon) \right\}, \max_{i\notin Q} \left\{ \frac{p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon}{m_i} \right\}, \max_{i\in Q} \left\{ \frac{p_i + 2\epsilon + \beta + \delta_{i-1,i}}{m_i} \right\} \right\}, \tag{8.1}$$

where $i_{j,|\mathcal{R}_j|+1} = i_{j1}$ and $\lambda = LCM_{1\le j\le r}[\lambda_j]$.

Proof. Activity $A_{i_{j_k}\gamma\eta}$ includes the unloading of machine $M_{i_{j_k}\gamma}$ (ϵ), moving to machine $M_{i_{j_k}+1,\eta}$ $(\delta_{i_{j_k}, i_{j_k}+1})$, and loading $M_{i_{j_k}+1,\eta}$ (ϵ). Immediately after performing activity $A_{i_{j_k}\gamma\eta}$, robot R_j travels to stage $i_{j_{k+1}}$ $(\delta_{i_{j_k}+1, i_{j_{k+1}}})$ to begin its next activity. The first argument of (8.1) represents the time for R_j to serve each stage i_{j_k}, $k = 1, \ldots, |\mathcal{R}_j|$, exactly once.

If robot R_j waits before it unloads machine $M_{i\gamma}$, $i \notin Q$, then the time between successive unloadings of $M_{i\gamma}$ is $p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon$ (it follows the same actions as those described in the proof of Theorem 8.1). The m_i machines of stage i are unloaded cyclically, so in the time between successive unloadings of $M_{i\gamma}$, the number of parts produced is m_i. Hence, the per unit cycle time for R_j is $(p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon)/m_i$.

If robot R_j waits before it unloads a machine $M_{i\gamma}$, $i \in Q$, then the time between successive unloadings of $M_{i\gamma}$ is $p_i + 2\epsilon + \beta + \delta_{i-1,i}$. Hence, the per unit cycle time for R_j is $(p_i + 2\epsilon + \beta + \delta_{i-1,i})/m_i$. ∎

COROLLARY 8.1 *For a robotic cell with parallel machines, r robots, and a fixed assignment of stages to robots, π_{LD} is optimal if*

$$\max\left\{\max_{i\notin Q}\left\{\frac{p_i+\delta_{i,i+1}+\delta_{i+1,i-1}+\delta_{i-1,i}+4\epsilon}{m_i}\right\},\right.$$
$$\left.\max_{i\in Q}\left\{\frac{p_i+2\epsilon+\beta+\delta_{i-1,i}}{m_i}\right\}\right\}\geq$$
$$\max_{1\leq j\leq r}\left\{\sum_{k=1}^{|\mathcal{R}_j|}(\delta_{i_{j_k},i_{j_k}+1}+\delta_{i_{j_k}+1,i_{j_{k+1}}}+2\epsilon)\right\}.$$

Proof. The proof follows from Theorem 8.1 and Theorem 8.2. ∎

COROLLARY 8.2 *For a robotic cell with parallel machines, r robots, constant travel-time δ, and a fixed assignment of stages to robots, π_{LD} is optimal if $p_i \geq \delta$, $i = 1, \ldots, m$.*

Proof. Corollary 8.1 implies that we need only to consider the case for which

$$\frac{T(\pi_{LD})}{\lambda} = 2\max_{1\leq j\leq r}|\mathcal{R}_j|(\delta+\epsilon).$$

From Theorem 8.1, if $p_i \geq \delta, \forall i$, then

$$\frac{T(\pi)}{k} \geq \max_{1\leq j\leq r}\{|\mathcal{R}_j|(\delta+2\epsilon)+|\mathcal{R}_j\setminus Q|\delta+$$
$$|\mathcal{R}_j\cap(Q\cup\{0\}|\delta\} = 2\max_{1\leq j\leq r}|\mathcal{R}_j|(\delta+\epsilon),$$

for all k-unit cycles π. Hence, π_{LD} is optimal. ∎

8.4 Scheduling by a Heuristic Dispatching Rule

The semiconductor equipment manufacturer mentioned in Section 8.1 currently uses a scheduling policy called *Longest Waiting Pair* (LWP). This is a heuristic dispatching scheme in which a centralized control computer determines each robot's next action at the completion of that robot's current action. Hence, it specifies only one activity at a time. This contrasts with cycles such as π_{LD}, in which all robot actions are specified before the cell's processing begins.

The Longest Waiting Pair policy is implemented as follows for a cell with one machine at each stage. For each part that has completed processing on its current machine, the control computer tracks how long

that part has been waiting to be unloaded. For each empty machine, the control computer tracks how long that machine has been waiting for its next part. The waiting time of each part is added to the waiting time of its destination machine. The pair with the largest sum of waiting times when the corresponding robot completes an action is designated the longest waiting pair, and the robot's next action is to move that part to its destination machine.

This scheme can be formalized for a cell with $m_i \geq 1$ machines at stage $i \in M$ as follows. For each machine M_{ih}, let c_{ih} be the time at which machine M_{ih} completes the processing of its current part, $i \in M$, $h \in \{a, \ldots, \alpha(m_i)\}$. When M_{ih} is unloaded, c_{ih} is set to ∞; $c_0 = 0$ throughout all processing. Let $c_i = \min_{a \leq h \leq \alpha(m_i)}\{c_{ih}\}$. Let u_{ih} be the time at which M_{ih} is unloaded, $i \in M$, $h \in \{a, \ldots, \alpha(m_i)\}$. When M_{ih} is loaded, u_{ih} is set to ∞; $u_{m+1} = 0$ throughout all processing. Let $u_i = \min_{a \leq h \leq \alpha(m_i)}\{u_{ih}\}$. When robot R_j completes an action, let $i' \in \mathcal{R}_j$ be a value for which $c_i + u_{i+1}$ is minimal. In the case of ties, the stage furthest downstream, i.e., with the largest index, is chosen. Robot R_j unloads a machine in stage i' for which $c_{i'h}$ is minimal (break ties arbitrarily), takes that part to a machine in stage $i'+1$ for which $u_{i'+1,h}$ is minimal (break ties arbitrarily), and loads that machine.

8.5 Computational Results

Simulations were used (Geismar et al. [64]) to compare the performance of the Longest Waiting Pair (LWP) robot scheduling scheme to π_{LD} for the 18-stage robotic cell with parallel machines and three robots described in Section 8.1. They were run for each of three different configurations of the 18-stage cell. The configurations have different sets of values for the number of parallel machines m_i at each stage, $i = 1, \ldots, 18$. For each configuration, simulations were run for fifteen different data sets. Within each set are 100 problem instances, each of which consists of the processing times for each of the 18 stages. For a given data set, each stage's 100 processing times were generated as realizations of a normal random variable with a fixed mean and standard deviation. The different data sets were created by varying the parameters − five sets of means with three sets of standard deviations for each mean.

For each configuration, five different sets of means were generated as follows: the first set is called the primary means (PM). Other sets were created by taking one-half of each mean (PM*0.5), 80% of each (PM*0.8), 120% of each (PM*1.2), and 150% of each mean (PM*1.5). For each set of means, three different sets of standard deviations were used. In the first, second, and third, respectively, the standard deviation of each stage's processing time was one-half, one-fourth, and one-eighth of that stage's mean. A simulation run using LWP was made for each problem instance.

For each of the 15 data sets of each configuration, the average of the percentage improvement realized from using π_{LD} rather than LWP was computed. These values are presented in Table 8.1. When interpret-

	Std.	Means (%)				
Cnfg.	**Devs.**	**PM*0.5**	**PM*0.80**	**PM**	**PM*1.2**	**PM*1.5**
	mean/2	7.34	4.81	3.12	2.73	2.49
1	**mean/4**	9.62	4.94	3.89	3.88	2.44
	mean/8	10.85	5.67	4.03	2.99	2.62
	mean/2	8.75	12.74	10.53	7.72	5.89
2	**mean/4**	7.21	12.74	12.30	10.69	7.75
	mean/8	6.69	9.53	12.62	9.49	6.96
	mean/2	11.04	9.73	8.29	6.45	5.75
3	**mean/4**	13.33	9.04	6.37	5.58	4.16
	mean/8	18.05	9.19	7.01	5.66	4.55

Table 8.1. Average Improvements from Using π_{LD} Instead of LWP.

ing these data, we should note that the condition of Corollary 8.1 is satisfied (and, hence, π_{LD} is optimal) in 87% of the problem instances. Table 8.2 compares the per-unit cycle times of LWP and π_{LD} to the lower bound of Theorem 8.1. These results testify to the quality of this cycle by demonstrating its optimality for a variety of instances. An example in which π_{LD} does not achieve this lower bound occurs in Configuration 2, PM*0.5: $T(\pi_{LD})/\lambda = 33$, but the lower bound of Theorem 8.1 is $T(\pi)/k \geq 28, \forall\pi$. It is not known if a k-unit cycle π^* for which $T(\pi^*)/k < 33$ exists, nor has it been proven that one does not exist.

Cnfg.	Process times	Per-unit cycle time lwrbnd	π_{LD}	LWP	π_{LD} % above lwrbnd	LWP % above lwrbnd
	PM*0.5	47	47	51	0.00%	8.51%
	PM*0.8	71	71	74	0.00%	4.23%
1	PM	87	87	90	0.00%	3.45%
	PM*1.2	103	103	106	0.00%	2.91%
	PM*1.5	127	127	130	0.00%	2.36%
	PM*0.5	28	33	34	17.86%	21.43%
	PM*0.8	30	33	39.47	10.00%	31.57%
2	PM	36	36	52	0.00%	44.44%
	PM*1.2	42	42	64	0.00%	52.38%
	PM*1.5	51	51	68	0.00%	33.33%
	PM*0.5	40	40	46.96	0.00%	17.40%
	PM*0.8	60	60	67.75	0.00%	12.92%
3	PM	73	73	80.5	0.00%	10.27%
	PM*1.2	86	86	93.97	0.00%	9.27%
	PM*1.5	106	106	110	0.00%	3.77%

Table 8.2. Comparison of Per-Unit Cycle-Times for LWP and π_{LD} to the Lower Bound of Theorem 8.1.

π_{LD}'s superiority can in part be attributed to it being a predefined schedule. In the instances for which it is known to be optimal ($T(\pi_{LD})/\lambda = (p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon)/m_i,\ i \notin Q$, or $T(\pi_{LD})/\lambda = (p_i + 2\epsilon + \beta + \delta_{i-1,i})/m_i,\ i \in Q$), the appropriate robot R_j, $j \in \{1, \ldots, r\}$, is waiting at machine $M_{i\gamma}$, $\gamma \in \{a, \ldots, \alpha(m_i)\}$, when it completes processing at time $t_{i\gamma}$. Thus, R_j can unload $M_{i\gamma}$ at time $t_{i\gamma}$. Under the LWP schedule, R_j is never waiting at $M_{i\gamma}$ when it completes processing. At best, at time $t_{i\gamma}$, R_j will begin to travel to $M_{i\gamma}$ for unloading. Thus, the earliest that R_j can unload $M_{i\gamma}$ is at time $t_{i\gamma} + \delta_{hi}$, $h \in \bar{\mathcal{R}}_j$. Hence, if the condition of Corollary 8.1 is satisfied, then

$$\frac{T(LWP)}{k} - \delta_{hi} \geq \frac{T(\pi_{LD})}{\lambda},\ h \in \bar{\mathcal{R}}_j,\ j \in \{1, \ldots, r\}.$$

For cases in which the condition of Corollary 8.1 is not satisfied, the robot is never waiting at a machine when it completes processing for either schedule. Consider, however, that in the simulations, we have $p_i > \delta_{jk},\ i = 1, \ldots, 18,\ 0 \leq j, k \leq 19$. Therefore, by Corollary 8.2, if

this were a constant travel-time cell, then π_{LD} would be optimal. Since the travel-time metric is a generalization of constant travel-time, and the range of values for travel times is small in the simulations, it is not surprising that π_{LD} works well also in these instances.

8.6 Applying an LCM Cycle to Implemented Cells

Simulations were also run using both the Longest Waiting Pair (LWP) schedule and the LCM cycle π_{LD} for four cells that are already being used in industry. Each cell has three robots, and the number of stages in the cells varies from 10 to 15. In two of the cells, physical constraints require that there be a stage that is used only to transfer a part from robot R_2 to robot R_1 before robot R_1 places the part into the output device. This stage is modeled just like the other stages, but its machines' processing times are zero. Another difference in these implementations is the travel times of the robots: traveling between two machines takes longer with a part than without one. This is done to protect the parts, and is modeled by choosing appropriate values for the loading and unloading times.

For these four cells, the percentage improvement in throughput gained from using π_{LD} rather than LWP ranged from 6.23% to 12.02%, with an average of 9.1%. In only one cell (Cell 1) did π_{LD} achieve the lower bound of Theorem 8.1 by satisfying the conditions of Corollary 8.1. The largest deviation from this lower bound was 9.05% for Cell 2. Interestingly, this is also the cell for which π_{LD} showed the greatest improvement over LWP, which also suggests that the lower bound may not be achievable in this case. At present it is not known if there is a sequence that is more efficient for this cell. A summary of these results is given in Table 8.3.

Cell	Number of stages	Percentage improvement π_{LD} v. LWP	Percentage π_{LD} is above lower bound
0	11	9.26	1.49
1	10	6.23	0.00
2	14	12.02	9.05
3	15	8.88	7.11

Table 8.3. Comparison of π_{LD} to LWP for Implemented Cells.

Chapter 9

NO-WAIT AND INTERVAL ROBOTIC CELLS

Most of the discussion in the previous chapters has been for free-pickup cells. Cells with two other pickup criteria have been investigated in the literature – no-wait cells and interval cells (see Chapter 2). This chapter briefly discusses cyclic production in these cells. In Section 9.1, we discuss no-wait cells. Section 9.2 summarizes the research for interval cells.

9.1 No-Wait Robotic Cells

In no-wait cells, a part must be removed from machine M_i, $i \in M$, and transferred to machine M_{i+1} as soon as M_i completes processing that part. Such conditions are commonly seen in steel manufacturing or plastic molding, where the raw material must maintain a certain temperature, or in food canning to ensure freshness (Hall and Sriskandarajah [77]). Examples of no-wait cells also include chemical processing tank lines, such as those used for manufacturing printed circuit boards or airplane wings (Song et al. [145]). We first consider cells producing a single part-type and then discuss multiple-part-type production.

Consider problem $RF_m|(\textit{no-wait,E,cyclic-1})|\mu$. To satisfy the no-wait constraint, the robot must be available at each machine by the time that machine completes its processing of a part. More formally, each part is unloaded from machine M_i (i.e., activity A_i begins) exactly Z_i time units after it enters the cell, where $Z_i = \sum_{k=1}^{i}(p_k + \delta_{k-1,k} + 2\epsilon)$, $i = 1, \ldots, m$. Hence, a schedule is completely determined by the input

times of the parts. These input times must be carefully selected to ensure feasibility for a cycle; in no-wait cells, the assumption of always using active schedules (Chapter 3) does not apply. In fact, under the no-wait constraint there are certain cells for which a particular 1-unit cycle has no feasible schedule. For example, consider a three-machine cell running cycle $\pi_4 = (A_0, A_3, A_1, A_2)$. After loading M_1, the robot must travel to M_3 (which requires time δ_{13}), perform A_3 ($\delta_{34} + 2\epsilon$), and return to M_1 (δ_{41}) by the time M_1 finishes processing. If $p_1 < \delta_{13} + \delta_{34} + \delta_{41} + 2\epsilon$, then π_4 is infeasible for this cell.

The no-wait constraint has been exploited to create a polynomial-time algorithm to find an optimal solution for $RF_m|(\textit{no-wait,E,cyclic-1})|\mu$. We present this algorithm, developed by Levner et al. [111]. The approach is commonly referred to as a *sieve method* or a *prohibited intervals* approach. We later look at its extensions to 2-unit cycles, to cells processing two part-types, and to multiple-robot cells, and then at other results for processing multiple part-types. We know of no studies of no-wait cells with parallel machines or dual-gripper robots.

Because the cycle time T in a 1-unit cycle is the length of the interval between successive parts entering the cell, the definition of Z_i implies that $Y_i = Z_i \pmod T$ is the time that passes from the start of the cycle until activity A_i begins. Hence, each feasible value of T implies a particular cycle.

EXAMPLE 9.1 Let $m = 4$, $p = (18, 24, 15, 18)$, $\delta_{i,j} = 1$ for $0 \le i < j \le 4$, $\epsilon = .5$. These data imply $Z_0 = 0$, $Z_1 = 20$, $Z_2 = 46$, $Z_3 = 63$, and $Z_4 = 83$. If $T = 33$, then $Y_0 = 0$, $Y_1 = 20$, $Y_2 = 13$, $Y_3 = 30$, and $Y_4 = 17$; so cycle $(A_0, A_2, A_4, A_1, A_3)$ has cycle time $T = 33$. Figure 9.1 presents a Gantt chart for this example in which each row represents a part's schedule, each segment represents processing on a machine, and the circled activities comprise one iteration of the cycle.

For any given no-wait cell, our task is to find the smallest feasible 1-unit cycle time T^* and a cycle that achieves this time. We start by considering constraints imposed by the robot's movement times.

For a cycle to be feasible, the interval between the starting times of a cycle's consecutive activities A_i and A_j must be large enough for the robot to perform its required actions ($\delta_{i,i+1} + 2\epsilon + \delta_{i+1,j}$). Therefore, for any pair of indices (i, j) such that $0 \le j < i \le m$, the triangle inequality

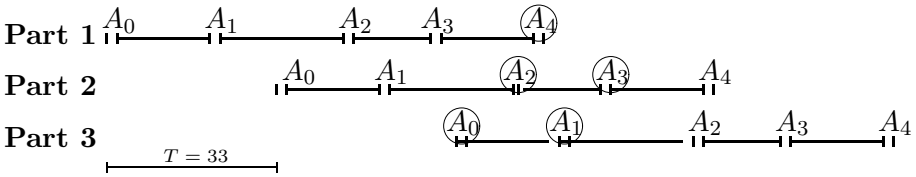


Figure 9.1. Gantt Chart for Cycle $(A_0, A_2, A_4, A_1, A_3)$ in Example 9.1.

implies that

$$\begin{aligned} &\text{either} \quad Y_j \geq Y_i + \delta_{i,i+1} + \delta_{i+1,j} + 2\epsilon, \\ &\text{or} \quad Y_j \leq Y_i - \delta_{j,j+1} - \delta_{j+1,i} - 2\epsilon. \end{aligned}$$

From the definition of Y_i, we have $Y_i = Z_i - s_i T$ for some integer $s_i \in [0, m-1]$. Hence, the above pair of inequalities becomes

$$\begin{aligned} &\text{either} \quad (s_i - s_j)T \geq Z_i - Z_j + \delta_{i,i+1} + \delta_{i+1,j} + 2\epsilon, \\ &\text{or} \quad (s_i - s_j)T \leq Z_i - Z_j - \delta_{j,j+1} - \delta_{j+1,i} - 2\epsilon. \end{aligned}$$

The right-hand side of each inequality in this pair contains known data. On the left-hand side is the unknown T. Although the values of s_i and s_j are uniquely determined by any triple (Z_i, Z_j, T), note that this exclusive disjunction of inequalities must hold true for all values of $(s_i - s_j) \in [0, m-1]$, because the cycle is repeated *ad infinitum*. Thus, we have the following extended system of alternative inequalities in which the only unknown is T:

$$\begin{aligned} T &\geq Z_i - Z_j + \delta_{i,i+1} + \delta_{i+1,j} + 2\epsilon, \quad \text{or} \\ T &\leq Z_i - Z_j - \delta_{j,j+1} - \delta_{j+1,i} - 2\epsilon; \\ 2T &\geq Z_i - Z_j + \delta_{i,i+1} + \delta_{i+1,j} + 2\epsilon, \quad \text{or} \\ 2T &\leq Z_i - Z_j - \delta_{j,j+1} - \delta_{j+1,i} - 2\epsilon; \\ &\vdots \\ (m-1)T &\geq Z_i - Z_j + \delta_{i,i+1} + \delta_{i+1,j} + 2\epsilon, \quad \text{or} \\ (m-1)T &\leq Z_i - Z_j - \delta_{j,j+1} - \delta_{j+1,i} - 2\epsilon; \\ &\forall (i,j) \text{ such that } \ 0 \leq j < i \leq m. \end{aligned}$$

There are $(m-1)m(m+1)/2$ pairs of alternative inequalities. Each pair defines an open interval into which T cannot fall. This leads directly

to an algorithm for finding an optimum cycle time and a corresponding cycle.

Algorithm No-Wait

Step 1: Let

$$\begin{aligned} T'_{ij} &= Z_i - Z_j - \delta_{j,j+1} - \delta_{j+1,i} - 2\epsilon, \\ T''_{ij} &= Z_i - Z_j + \delta_{i,i+1} + \delta_{i+1,j} + 2\epsilon, \end{aligned}$$

$\forall (i,j)$ such that $0 \le j < i \le m$.

Step 2: Define

$$I_k = [0, \max_{1 \le i \le m} \{p_i + \quad \delta_{i,i+1} + \quad \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon\}) \cup \left\{ \left(\frac{T'_{ij}}{k}, \frac{T''_{ij}}{k} \right) : 0 \le j < i \le m \right\},$$

for $k = 1, \ldots, m-1$.

Step 3: For each $k = 1, \ldots, m-1$, sort the intervals of I_k in non-decreasing order of their left endpoints, merging the intersecting intervals.

Step 4: Construct the ordered set $I = I_1 \cup I_2 \cup \cdots \cup I_{m-1}$, merging the intersecting intervals.

Step 5: Choose the right end of the first interval in I. This value is T^*, the minimum feasible cycle time.

Step 6: Compute $Y_i = Z_i \mod T^*$, $i = 0, \ldots, m$. Arrange the Y_i's in ascending order to determine an optimal cycle.

EXAMPLE 9.2 Recall the data for Example 9.1: $m = 4$, $p = (18, 24, 15, 18)$, $\delta_{i,j} = 1$ for $0 \le i < j \le 4$, $\epsilon = .5$, so $Z_0 = 0$, $Z_1 = 20$, $Z_2 = 46$, $Z_3 = 63$, and $Z_4 = 83$. First find the end points for the forbidden

intervals:

$$\begin{array}{llll} T'_{01} = 17, & T'_{02} = 43, & T'_{03} = 60, & T'_{04} = 80 \\ T''_{01} = 23, & T''_{02} = 49, & T''_{03} = 66, & T''_{04} = 86 \\ & T'_{12} = 23, & T'_{13} = 40, & T'_{14} = 60 \\ & T''_{12} = 29, & T''_{13} = 46, & T''_{14} = 66 \\ & & T'_{23} = 14, & T'_{24} = 34 \\ & & T''_{23} = 20, & T''_{24} = 40 \\ & & & T'_{34} = 17 \\ & & & T''_{34} = 23 \end{array}$$

Next, define and combine the intervals:

$$\begin{aligned} I_1 &= \{[0,29),(34,40),(40,49),(60,66)\} \\ I_2 &= \{[0,29),(30,33)\},\ I_3 = \{[0,29)\},\ \text{ so} \\ I &= \{[0,29),(30,33),(34,40),(40,49),(60,66)\}, \end{aligned}$$

and $T^* = 29$. Hence, $Y_0 = 0$, $Y_1 = 20$, $Y_2 = 17$, $Y_3 = 5$, and $Y_4 = 25$; so cycle $(A_0, A_3, A_2, A_1, A_4)$ is an optimal cycle with cycle time $T^* = 29$.

THEOREM 9.1 *Algorithm No-Wait finds an optimal 1-unit cycle and its cycle time in a no-wait cell in time* $O(m^3 \log m)$(Levner et al. [111]).

Proof. That algorithm No-Wait finds an optimum 1-unit cycle and its cycle time follows directly from the preceding discussion. We now consider its complexity step by step. In Step 1, there can be at most $m(m+1)/2$ values each for T'_{ij} and T''_{ij}. Hence, Step 1 is $O(m^2)$ and Step 2 is $O(m^3)$. Sorting $m(m+1)/2$ intervals can be performed in $O(m^2 \log m)$, so Step 3 is $O(m^3 \log m)$. For Step 4, observe that merging two sets of size k is $O(k)$. We first pairwise merge I_i and I_{i+1} for $i = 1, 3, \ldots, m-1$, which requires time $(m/2)O(m^2) = O(m^3)$. We now have $m/2$ sets of at most $2m^2$ intervals each. Pairwise merging these is $(m/4)O(m^2) = O(m^3)$. It follows that each successive pairwise merge will be $O(m^3)$, and there will be $O(\log m)$ rounds of merges, so Step 4 is $O(m^3 \log m)$. Step 5 is $O(1)$, and Step 6 is $O(m \log m)$. Hence, Algorithm No-Wait is $O(m^3 \log m)$. ∎

Kats and Levner [97] adapt this algorithm to find an optimal 1-unit cycle in a no-wait cell with multiple robots. The implementation is very similar to that of the single robot case and, hence, has the same

complexity. By using a similar set of alternative inequalities, Agnetis [2] shows that for $m = 3$, an optimal cycle produces either one unit or two units and provides a constant-time algorithm to find such an optimal cycle. Agnetis' conjecture, that for an m-machine no-wait cell there exists an optimal cycle that produces $m-1$ or fewer units, remains open. However, this has been proven for certain cases in balanced ($p_i = p, \forall i$) no-wait cells (Mangione et al. [119]). The corresponding statement for free-pickup cells with either an additive or a constant travel-time metric is invalid (Brauner [17]).

Che et al. [31] solve $RF_m|(\textit{no-wait,E,cyclic-2})|\mu$ with an extension of algorithm No-Wait that has time complexity $O(m^8 \log m)$. This algorithm is a refined version of an earlier geometric algorithm due to Levner et al. [112]. The complexity is much greater because there are forbidden intervals both for the cycle time T and for T_1, where parts are introduced to the cell at times $0, T_1, T, T+T_1, 2T, 2T+T_1, \ldots$, and because the forbidden intervals for T_1 depend on the value of T. A recent improvement by Chu [34] results in a $O(m^5 \log m)$ algorithm. Chen et al. [30] propose a branch-and-bound algorithm for $RF_m|(\textit{interval,E,cyclic-k})|\mu$. Two algorithms for a general k-unit cycle ($k \geq 2$) are each exponential in k. Kats et al. [98] develop a sieve-based algorithm with time complexity $O(m^3 k n^k)$, where $n = Z_{m+1} + \delta_{m+1,0} - \max_i\{p_i + \delta_{i-1,i} + \delta_{i,i+1} + \delta_{i+1,i-1} + 4\epsilon\}$ is the size of the range of feasible values for the length of a 1-unit cycle. Che et al. [30] develop a branch-and-bound procedure in which each node is solved by a linear program that is equivalent to the cycle time evaluation problem in a bi-valued graph. Unlike sieve-based algorithms, this approach does not require integer data, and may produce non-integer cycle times. Its time complexity is $O(k^2 m^{6+3k(k-1)/2})$.

Studies of cells producing multiple part-types consider only CRM (Chapter 6) sequences. The algorithm of Che et al. [31] finds the best 2-unit cycle for two different part-types if the MPS is (P_1, P_2). For a general MPS and $m = 2$, Agnetis [2] shows that the part scheduling problem can be solved using the Gilmore-Gomory algorithm. For $m = 3$, Agnetis and Pacciarelli [3] show that the part scheduling problem for the CRM sequence with $\pi_1 = (A_0, A_1, A_2, A_3)$ is trivial and that an optimal part schedule for the CRM sequence with $\pi_4 = (A_0, A_3, A_1, A_2)$ can be found by the Gilmore-Gomory algorithm. Whether or not a *feasible* part

schedule exists for the CRM sequences with $\pi_3 = (A_0, A_1, A_3, A_2)$ and $\pi_5 = (A_0, A_2, A_3, A_1)$ can be determined via the Gilmore-Gomory algorithm. However, the complexity of finding an *optimal* part schedule is an open problem. For MPS cycles corresponding to the CRM sequences with $\pi_2 = (A_0, A_2, A_1, A_3)$ and $\pi_6 = (A_0, A_3, A_2, A_1)$, the problem of finding a feasible part schedule is strongly NP-complete (Agnetis and Pacciarelli [3]).

9.2 Interval Pick-up Robotic Cells

In interval robotic cells, each stage has a specific interval of time − a processing time window − for which a part can be processed on that stage. This problem, also referred to as the hoist scheduling problem, often arises in electroplating and chemical industries. In the United States, there are more than 4,000 electroplating manufacturers who produce various industrial connectors, printed circuit boards, and switches used in telecommunications hardware (Lee at al. [105]). In typical electroplating applications, printed circuit boards are to be placed in a series of tanks with different solvents. Each tank has a specific interval of time for which a board can remain immersed in it.

In general, if $[a_i, b_i]$ is the processing time window at machine $M_i, i \in M$, then the interval-pickup criterion requires that a part be processed for a_i time units on machine M_i, and that it be transferred to machine M_{i+1} within $(b_i - a_i)$ time units after its completion of processing on M_i.

To our knowledge, all the studies performed thus far are for interval cells producing a single part-type. Problem $RF_m|(\textit{interval,E,cyclic-1})|\mu$ is strongly NP-hard (Livshits et al. [115], Crama and van de Klundert [41]). Since the free-pickup criterion is a special case of the interval-pickup criterion, this also follows from a result of Brauner et al. [24] that establishes the strong NP-hardness of $RF_m|(\textit{free,E,cyclic-1})|\mu$. It has to be noted, however, that the techniques used for proving these results are explicitly limited to the determination of an optimal l-unit cycle. Hence, the complexity of determining an optimal cyclic solution (i.e., problem $RF_m|(\textit{interval,E,cyclic-k})|\mu$) remains open, as no optimal sequence may be given by a l-unit cycle. We now briefly review the literature on solution methods for the interval robotic cell problem and closely related

variants. Problem $RF_m|(interval,E,cyclic\text{-}1)|\mu$, together with a linear programming formulation of the cycle time subproblem, was introduced independently by Livshits et al. [115] and by Phillips and Unger [130]. Phillips and Unger [130] formulate the problem as an integer linear program and report on results from using a commercial software package. Kats [93], Lei and Wang [108], Armstrong et al. [5], Shapiro and Nuttle [144], and Chen et al. [33] propose branch-and-bound procedures for interval cells and various extensions. Rochat [134] solves the problem using a genetic algorithm.

As in free-pickup cells, a natural question for interval cells is whether an optimal 1-unit cycle is superior to every nontrivial k-unit cycle, $k \geq 2$. Using a real-world example with 12 machines, Lei and Wang [108] show that the throughput of an optimal 2-unit cyclic solution is strictly better than that of an optimal 1-unit cyclic solution. Song et al. [145] and Kats et al. [98] report similar findings. A brief summary of known results for both no-wait and interval cells is given in Table 9.1. For interval cells,

Problem	Complexity Status	Solution Approach	
		Algorithm Type	References
$RF_m\|(no\text{-}wait,E,cyclic\text{-}1)\|\mu$	$O(m^3 \log m)$ [111]	Sieve Method	[111]
$RF_m\|(no\text{-}wait,E,cyclic\text{-}2)\|\mu$	$O(m^8 \log m)$ [33]	Sieve Method	[33]
$RF_m\|(no\text{-}wait,E,cyclic\text{-}k)\|\mu$	Open	Branch-and-Bound	[30]
$RF_m\|(interval,E,cyclic\text{-}1)\|\mu$	Strongly NP-hard [115, 41]	Branch-and-Bound	[5, 33, 93] [108, 144]
$RF_m\|(interval,E,cyclic\text{-}k)\|\mu$	Open	Branch-and-Bound	[145]

Table 9.1. Summary of Results for No-Wait and Interval Cells.

the complexity status of obtaining an optimum cyclic solution under the Euclidean travel-time metric (i.e., $RF_m|(interval,E,cyclic\text{-}k)|\mu$) remains open since all such solutions could potentially come from k-unit cycles, $k \geq 2$. A similar observation applies to no-wait cells as well.

Chapter 10

OPEN PROBLEMS

We now summarize some open problems in the field of robotic cell sequencing and scheduling.

10.1 Simple Robotic Cells

As mentioned in Chapter 2, at least nine different types of simple robotic cells with single-gripper robots have been studied in the literature, depending on the pickup criterion (no-wait, interval, or free-pickup) and the travel-time metric (additive, constant, or Euclidean). Recall our discussion of Chapter 3 where the cell was presented as a dynamic system completely described by a finite set $\mathcal{F}$ of feasible states. Furthermore, by definition, a k-unit cycle has $k(m+1)$ distinct states. It follows that an upper bound on the number of parts produced by any throughput maximizing cycle is

$$\omega = \frac{|\mathcal{F}|}{(m+1)}.$$

Let $\mathcal{C}_k, k = 1, ..., \omega$, denote the class of all k-unit cycles. Then a k-unit cycle C_k^* with $T(C_k^*) = \min_{C_k \in \mathcal{C}_k} T(C_k)$ can be executed indefinitely to obtain a k-unit cyclic solution with maximum throughput. A cyclic solution with the cycle time $T(C^*) = \min_{k=1,...,\omega} T(C_k^*)$ then maximizes the throughput over all cyclic solutions.

Thus, from an algorithmic point of view, the following problems become relevant for such cells:

1. **Problems k-OPT**: For additive (resp., constant, Euclidean) travel-time cells, obtain algorithms to find an optimal k-unit cycle for a given k, where $1 \leq k \leq \omega$.

2. **Problem OPT**: For additive (resp., constant, Euclidean) travel-time cells, obtain an algorithm to find a k-unit cycle that maximizes throughput over all cyclic solutions.

For each of the nine types of cells, the complexity of Problem k-OPT has been resolved for $k = 1$ (see Chapter 3). We start our discussion of specific open problems by mentioning the following two points.

- First, we would like to emphasize the need for efficient and easy-to-implement algorithms for solving these problems. Since the number of cyclic solutions is finite, a complete enumeration of such solutions (or any other method that requires enumerating a prohibitively large search space) will provide a valid answer to these problems. However, such methods do not provide a satisfactory resolution as they cannot be implemented in practice within reasonable time. In terms of computational complexity, we either want to obtain a polynomial-time algorithm for a problem or to show that it is NP-hard. While demonstrating that a problem is NP-hard does not completely rule out solution methods that solve instances of reasonable size to optimality, the use of heuristics to obtain approximate solutions becomes much more acceptable in such a case.

- Second, note that the most significant among these problems is Problem OPT, the problem of obtaining a cyclic schedule with maximum throughput. While solving Problems k-OPT, $k = 1, 2, ..., \omega$, automatically solves Problem OPT, it may not be necessary to do so. Indeed, for all the special cases in which problem OPT has been solved, the result follows without analyzing the individual problems k-OPT, $k = 1, 2, ..., \omega$. Over the past decade, a significant amount of research in the scheduling of robotic cells has, directly or indirectly, focused on answering these two problems.

For free-pickup cells with the additive and constant travel-time metrics, no algorithmic results are available for Problems k-OPT, $k \geq 2$. However, we do know that these problems are relevant, since examples

exhibiting the throughput of an optimal 2-unit cycle to be strictly better than the throughput of an optimal 1-unit cycle have been constructed. In other words, there exist instances of cells in which $T(C_2^*) < T(C_1^*)$. For additive travel-time (resp., constant travel-time) cells, Brauner and Finke [18, 22] (resp., Dawande et al. [45]) showed such instances (see Chapter 3). For interval robotic cells, problems $RF_m|(\textit{interval,A,cyclic-1})|\mu$ and $RF_m|(\textit{interval,E, cyclic-1})|\mu$ are NP-hard (Crama and van de Klundert [41]). No results have been published for interval cells with constant travel time. For Euclidean no-wait cells, polynomial algorithms to find an optimal 1-unit cycle (problem $RF_m|(\textit{no-wait,E,cyclic-1})|\mu$) and an optimal 2-unit cycle (problem $RF_m|(\textit{no-wait,E,cyclic-2})|\mu$) have been developed (Levner et al. [111], Che et al. [31], Chu [34]).

For Problem OPT, two different types of results are available: (i) those that solve the problem under specific conditions on the problem data, and (ii) those that provide a bound on the optimal throughput in terms of the throughput of an optimal 1-unit cycle. Under different conditions on the problem data, several results use the following two-step procedure to solve Problem OPT.

1 Obtain an upper bound on the maximum throughput over all cyclic solutions. For a k-unit cycle C_k, this is typically done by estimating a lower bound on the per-unit cycle time $T(C_k)/k$. Here, the analysis uses quantities such as the minimum number of robot moves involved, the number of loadings and unloadings, and the minimum total processing time required.

2 Provide a cyclic solution that achieves the upper bound (derived in Step 1 above) on the throughput, thus establishing its optimality.

Examples of such results are Lemmas 3.4 and 3.6 and Corollary 3.7, which establish the optimality of the 1-unit cycle $\pi_D = (A_0, A_m, A_{m-1}, \ldots, A_1)$ over all k-unit cycles.

The second type of results for Problem OPT estimate the gap between the throughput of an optimal cyclic solution and an optimal 1-unit cyclic solution. There are at least two reasons that motivate the estimation of this gap. (i) On the one hand, if the gap is very small, it can be used to devise new search methods that start with an optimal 1-unit solution and seek to improve its throughput by local search. Here, the hope is

that the small gap can be closed by a structured, limited search leading to an efficient algorithm. On the other hand, a large gap indicates that fundamentally new methods (from those employed for finding an optimal 1-unit cycle) are likely to be required to obtain an optimal cyclic solution. (ii) The gap is useful to assess the benefit of an optimal cyclic solution over that of an optimal 1-unit cyclic solution. This can be a significant insight, especially in practice. For additive cells, Crama and van de Klundert [40] show that the throughput of an optimal 1-unit cycle is at least 1/2 times that of an optimal cyclic solution. Geismar et al. [60, 62] improved this ratio first to 2/3, and then to 7/10. For constant (resp., Euclidean) cells, the best known ratio is 2/3 (resp., 1/4) (Geismar et al. [60]; see Chapter 3). For Eucliean cells, algorithm ECell (Chapter 3) offers a 1-unit cycle that has a performance guarantee of $2/3q$, where q is the ratio of the largest inter-machine travel time to the smallest. Clearly, these performance guarantees are too weak to pursue the local search mentioned in (i) above. However, it is important to note that none of these ratios has been shown to be tight. That is, for none of the classes of cells has it been shown that the above bounds are the largest possible for the ratio of the throughput of an optimal 1-unit cycle to the throughput of an optimal cyclic solution. Thus, a precise estimation of the gap remains open for the various classes of cells.

Another direction that has not been extensively explored is that of using mathematical programming formulations. Here, the idea is to express the problem of maximizing throughput of a cyclic solution as either a linear program or an efficient integer linear program (e.g., one in which the constraint matrix is totally unimodular; see e.g., Nemhauser and Wolsey [125]). For a given k-unit cyclic solution, the computation of its cycle time (and, hence, its throughput) can indeed be expressed as a linear program (Kumar [101], Kumar et al. [102]). However, attempts to formulate the problem of searching over all k-unit cyclic solutions have been unsuccessful so far.

We mention two other problems that have not been addressed in the literature, but might be helpful in understanding the structure of cyclic solutions in robotic cells.

- It is not known whether the maximum throughput of a cell over the class of k-unit cycles is an increasing function of k, $k \geq 1$. For exam-

ple, it is not known if the maximum throughput over all 3-unit cycles is at least as large as that over all 2-unit cycles. Indeed, nothing is known about the behavior of the maximum throughput with respect to k. However, if we define Problem $^{\leq}k$-OPT as that of determining a throughput maximizing cyclic solution over all l-unit cycles, $1 \leq l \leq k$, then it follows immediately that the throughput of an optimal solution of $^{\leq}k$-OPT is a non-decreasing function of k. However, there are no results on the rate of change of this increase. For instance, if the increase is decreasing in k (i.e., diminishing marginal increase), it might be reasonable to expect that Problem OPT attains its optimal at a relatively small value of k. For example, for Euclidean no-wait cells, Agnetis [2] has conjectured that an optimum cyclic solution is achieved by a k-unit cycle with $k < m$. Mangione et al. [119] prove this conjecture for certain balanced ($p_i = p, \forall i$) and regular additive no-wait cells. However, this property does not hold for free-pickup cells with either an additive or a constant travel-time metric (Brauner [17]).

- The dominance of cyclic solutions and most of the algorithmic results for robotic cells assume that all cell data is rational. This is a reasonable assumption in practice, and it greatly reduces the state space of the cell. Results for arbitrary real data, while not necessary for practical applications, seem to be much more challenging mathematically, and none are available in the literature.

Table 10.1 summarizes the complexity status of throughput optimization problems in simple robotic cells.

	Problems $RF_m\|(*,*,$**cyclic-k**$)\|\mu$		
travel-time metric $\rightarrow$ pickup criterion $\downarrow$	Additive	Constant	Euclidean
Free	$k = 1$ **[P]** **[open,** $k \geq 2$]	$k = 1$ **[P]** **[open,** $k \geq 2$]	**[NP-hard,** $k \geq 1$]
No-wait	$k = 1, 2$ **[P]** **[open,** $k \geq 3$]	$k = 1, 2$ **[P]** **[open,** $k \geq 3$]	$k = 1, 2$ **[P]** **[open,** $k \geq 3$]
Interval	**[NP-hard,** $k \geq 1$]	**[open,** $k \geq 1$]	**[NP-hard,** $k \geq 1$]

Table 10.1. Complexity Status of Simple Robotic Cell Problems.

10.2 Simple Robotic Cells with Multiple Part Types

We saw in Chapter 6 that two subproblems arise for multiple-part-type cells: part scheduling and robot move sequencing. Given a robot move sequence, finding an optimal part schedule is NP-hard for $m \geq 3$. The complementary problem of finding an optimal robot move sequence given a part schedule has not been studied for $m \geq 3$.

For cells that process multiple part-types with $m \geq 3$, all results in Chapter 6 concern CRM sequences (problem $RF_m|(free,A,MP,CRM)|\mu$). There has been no research on finding an optimal part schedule for a non-CRM MPS robot move sequence (problem $RF_m|(free,A,MP,cyclic\text{-}k)|\mu$) for $m \geq 3$. Such results will provide insight into how close an optimal MPS cycle formed by a CRM sequence is to an optimal MPS cyclic solution. This problem, however, is a generalization of the optimal k-unit cycle problem, which is also open.

10.3 Robotic Cells with Parallel Machines

The optimal cyclic solution has been found only for a special case ($p_i \geq \delta, \forall i$) in free-pickup constant travel-time cells with parallel machines (see Chapter 5). No studies have been published that address more general free-pickup constant travel-time cells or any free-pickup cells with additive travel times.

For cases in which finding an optimal robot move sequence in a simple robotic cell is known to be NP-hard (problems RF_m $|(free,E,cyclic\text{-}k)|\mu$, $RF_m|(interval,E,cyclic\text{-}k)|\mu$, and $RF_m|(interval,A,cyclic\text{-}k)|\mu$), the problems for cells with parallel machines are, of course, NP-hard. Finding efficient approximation algorithms remains an open question for these cases.

As discussed in Chapter 9, the problem of finding an optimal 1-unit cycle in no-wait simple robotic cells is polynomially solvable. We know of no work that addresses no-wait robotic cells with parallel machines (problem $RF_m(m_1, m_2, \ldots, m_m)|(no\text{-}wait,*,cyclic\text{-}k)|\mu$).

Finding an optimal schedule for processing multiple part-types in robotic cells with parallel machines (for any travel-time or pickup scheme (problem $RF_m(m_1, m_2, \ldots, m_m)|$ $(*,*,MP,cyclic\text{-}k)|\mu)$) is also a gener-

alization of the optimal k-unit cycle problem. Hence, it is currently an open question.

10.4 Stochastic Data

Throughout our discussion, we have assumed a deterministic setting, i.e., the processing times of the parts on the machines, the inter-machine travel times for the robot(s), and the loading and unloading times are all assumed to be known. Under the additional assumption of rationality of this data, a robotic cell is essentially a *finite-state* dynamic system. For maximizing the throughput of the cell, it is not necessary to consider "wasteful" robot actions such as unnecessary waiting at a location or moving to a location without performing at least one of the loading or unloading operations. Also, it is sufficient to define decisions regarding the robot's moves only at those epochs when the robot has just finished loading or unloading a part at a machine. It follows that it is sufficient to consider the cell's states at these epochs. As discussed in Chapter 2, a stochastic setting will typically require a continuous state space and continuous decision making over time. Ad-hoc dispatching schemes, such as the longest waiting pair scheme described in Chapter 8, are applicable in the stochastic setting; no mathematical analysis of such schemes is available.

Another issue concerns random failures of processing machines. Some manufacturers use parallel machines to provide redundancy. However, the following question remains largely open. Given each machine's rate of failure, the distribution of each machine's service and repair times, each machine's cost, and the distribution of each machine's processing time, what is the economically optimal number of machines to have at each stage? Such a calculation should also formulate the economic impact of improved throughput, i.e., cycle time comparison is a subproblem to the problem of finding an optimal number of redundant machines. Suri [150] provides some guidelines for addressing such questions.

10.5 Dual-Gripper Robots

Cyclic solutions for cells with a dual-gripper robot have not been studied for Euclidean travel-time (problem $RF_m^2|(*,E,\textit{cyclic-}k)|\mu$), interval cells (problems $RF_m^2|(\textit{interval},*,\ \textit{cyclic-}k)|\mu$), and no-wait cells (prob-

lems $RF_m^2|(no\text{-}wait,*,cyclic\text{-}k)|\mu)$. It would also be interesting to examine the use of dual-gripper robots in cells with multiple robots and parallel machines. Such a study could be based on the results in Geismar et al. [64].

10.6 Flexible Robotic Cells

Recent work has studied robotic cells that are open shops, rather than flowshops. In these cells, the operations can be performed in any order, and each machine can be configured to perform any of the operations. Geismar et al. [63] show that for $m = 3$ and $m = 4$, the largest productivity gain that can be realized by changing the assignment of operations to machines is $14\frac{2}{7}\%$. It is unknown whether this upper bound holds for $m \geq 5$.

10.7 Implementation Issues

We discuss two issues: the optimal use of local material handling devices, and the economic and performance trade-offs concerning schedules that revisit machines.

10.7.1 Using Local Material Handling Devices

Local material handling devices (LMHDs) can be used to increase throughput if the robot is the bottleneck. An LMHD transports a part from stage j to stage $j+1, j = 1, \ldots, m-1$, independently of the robot. If individual stages of a cell have buffers, then such devices can also be used to move completed parts from a machine to its buffer (see Section 4.8). If LMHDs are used in cells with parallel machines, then $m_j = m_{j+1}$, and each machine of stage j is linked to a unique machine of stage $j+1$. Furthermore, no stage can be linked to two stages: if stage j is linked to stage $j+1$, then stage $j-1$ is not linked to stage j, and stage $j+1$ is not linked to stage $j+2$. There are additional physical constraints (e.g., machine sizes, cell layout) which limit the number of links in a particular cell. For ease of exposition, for the remainder of this section we describe only a simple robotic cell.

LMHDs increase throughput by transferring a completed part as soon as possible and by reducing the robot's workload. If stages j and $j+1$ are linked, then the robot never performs activity A_j (unload M_j,

transport the part to M_{j+1}, load M_{j+1}). In a simple cell with only one LMHD link (from M_j to M_{j+1}), a 1-unit cycle is very similar to a 1-unit cycle for a cell with $m-1$ machines; it contains m activities $\{A_0, \ldots, A_{j-1}, A_{j+1}, \ldots, A_m\}$, and there are $(m-1)!$ possible cycles.

When the robot arrives at M_j to load a part, the linked pair of machines is either empty or contains one part. If the robot loads an empty pair, then the cell's operations and the calculations of its waiting times and cycle time are identical to those of an $(m-1)$-machine cell in which machine M_j^* is loaded and unloaded from different ports and $p_j^* = p_j + p_{j+1} + y_j$, where y_j is the time required for the LMHD to transfer the part from M_j to M_{j+1}. If the pair has one part when the robot arrives to load M_j, then M_j may still be processing the previous part. In this case, the robot must wait for M_j to complete processing and for the LMHD to transfer that part to M_{j+1}.

Another waiting time occurs if M_j completes processing, only to find M_{j+1} still occupied. In this case, the LMHD cannot transfer the part until the robot unloads M_{j+1}. Thus, a completed part waits on M_j until M_{j+1} is unloaded.

Kumar et al. [102] use simulation, linear programming, and genetic algorithms to demonstrate how LMHDs can improve throughput for a particular company's robotic cell. To our knowledge, there has been no general theoretical analysis to establish an optimal cycle for such an implementation or to determine the conditions which most favor that two machines be joined by an LMHD.

10.7.2 Revisiting Machines

Thus far, we have discussed techniques to improve throughput. We now examine a way to reduce cost without degrading throughput, i.e., to improve the performance-to-cost ratio.

In semiconductor manufacturing, certain stages, e.g., bake and chill, are repeated. If the machine that performs the bake at stage α also performs a bake of the same duration at stage β, an m-machine process can be performed by $m-1$ machines. This could provide significant savings in capital investment.

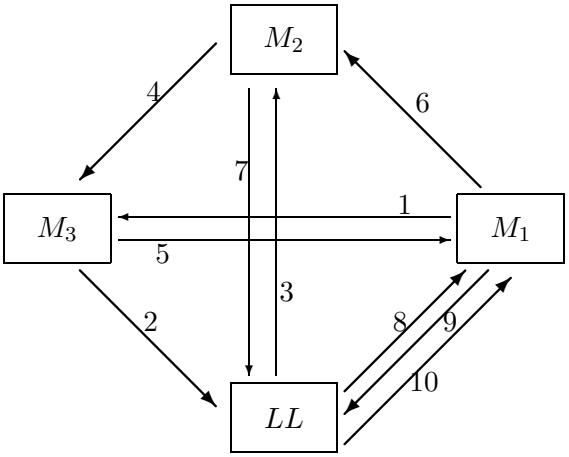


Figure 10.1. Machine Revisitation Sequence Using Load Lock for Intermediate Storage. Numbers Correspond to the Order of Robot Moves. Arcs 2, 4, 6, 8, 9, and 10 Represent Robot Moves with a Part.

To maintain throughput while reducing the number of machines, the processing time of the revisited machine M_α must be less than one half of the cell's other processing times: $p_\alpha \leq \min_{i \neq \alpha} p_i/2$. Furthermore, since a revisiting sequence requires more robot movements, δ and ϵ should be relatively small in comparison to $\max_{1 \leq i \leq m} p_i$, e.g., $\max_{1 \leq i \leq m} p_i + 3\delta + 4\epsilon \geq 2(m+1)(\delta + \epsilon)$ in a constant travel-time cell.

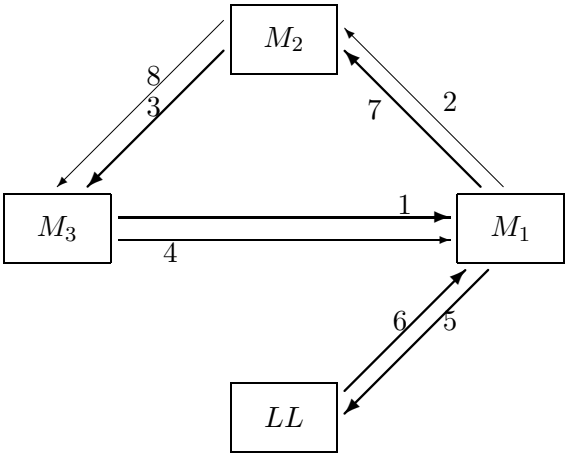


Figure 10.2. Machine Revisitation Sequence Without Using Load Lock for Intermediate Storage. Numbers Correspond to the Order of Robot Moves. Arcs 1, 3, 5, 6, and 7 Represent Robot Moves with a Part.

Perkinson et al. [129] implement such a revisiting scheme in a three-machine cell that is used for a four-stage process (problem $RF_4|(\textit{free,C, cyclic-1})|\mu$). The load lock (combined Input/Output device) is used as a buffer to store partially completed parts before their second visit to M_1 for processing stage 4. Their scheme, shown in Figure 10.1, is based on π_D (each machine is reloaded as soon as possible after it is unloaded) and has cycle time:

$$T_c(\pi_1) = \max\{p_1 + 10\delta + 12\epsilon, 2p_1 + 5\delta + 8\epsilon, p_2 + 3\delta + 4\epsilon, p_3 + 3\delta + 4\epsilon\}.$$

A particular implementation may not allow the use of the load lock as a buffer. It is still possible to revisit a machine in this case. An example of such a sequence is shown in Figure 10.2. Its cycle time is

$$T_c(\pi_2) = \max\left\{p_1 + 8\delta + 10\epsilon, 2p_1 + 5\delta + 8\epsilon, p_1 + p_2 + 5\delta + 8\epsilon, p_3 + 3\delta + 4\epsilon, \frac{2p_1 + p_2 + p_3 + 5\delta + 10\epsilon}{2}\right\}.$$

To our knowledge, there has been no thorough analysis of revisitation schemes to determine general guidelines or to evaluate the trade-off between performance and cost.

Appendix A

A.1 1-Unit Cycles

Sethi et al. [142] define 1-unit cycles in the context of single-part-type production and prove that there are exactly $m!$ potentially optimal 1-unit cycles in an m machine cell ((problem $RF_m|$(*free,A,MP,cyclic-1*)$|\mu$),). Here we discuss how to develop these cycles for $m = 2$, 3, and 4. In such a cell, we define $m + 1$ *basic activities* as follows:

M_i^-: Load a part on machine $M_i, i = 1, \ldots, m$.
M_m^+: Unload a finished part from machine M_m.

This appendix provides a brief tutorial on how to interpret a robot move cycle composed of these basic activities. A convenient point from which to begin the interpretation is with the unloading of a part from the last machine M_m, denoted by M_m^+. Note that M_m^+ implies that the next action of the robot is M_{m+1}^-, i.e., the robot moves to the output device O to drop the completed part. Consider the sequence ξ of basic activities from one occurrence of M_m^+ to the next such occurrence. Within ξ, there must be exactly one occurrence of each activity $M_1^-, M_2^- \ldots, M_m^-$. The machines that are occupied by parts at the time when M_m^+ occurs can be determined as follows. For $i = 1, \ldots, m-1$, machine M_i is occupied by a part when M_m^+ occurs if and only if M_{i+1}^- precedes M_i^- in ξ. This interpretation provides a starting point for the cycle. Since exactly which machines are occupied is known, the sequence of basic activities following M_m^+ uniquely determines the ordering of loading and unloading of the machines and, thus, also the robot movements between

activities. Examples for two- and three-machine robotic cells are given below.

Note that every basic activity must be carried out exactly once in a 1-unit cycle. Moreover, any two consecutive activities uniquely determine the robot moves between those activities. Therefore, a cycle can be uniquely described by a permutation of the above $m+1$ activities. The following are the 1-unit robot move cycles for $m = 2$.

A.1.1 1-Unit Cycles in Classical Notation

We describe 1-unit cycles for $m = 2, 3$, and 4, using the classical notation used by Sethi et al. [142].

$\pi_{1,2} = \{M_2^+, M_1^-, M_2^-, M_2^+\}$ $\pi_{2,2} = \{M_2^+, M_2^-, M_1^-, M_2^+\}$.

We now interpret these two cycles. In cycle $\pi_{1,2}$, we observe that following M_2^+, activity M_2^- does not precede M_1^-. Therefore, we know that machine M_1 is free when M_2^+ occurs. By contrast, in cycle $\pi_{2,2}$, activity M_2^- precedes M_1^-. Therefore, we know that machine M_1 is occupied by a part when M_2^+ occurs.

The six robot moves cycles for $m = 3$ can be developed from the above two cycles by first replacing the starting and ending activities by M_3^+, as shown below.

1. $\{M_3^+, M_1^-, M_2^-, M_3^+\}$
2. $\{M_3^+, M_2^-, M_1^-, M_3^+\}$.

Each of the last two cycles generates three cycles in a three machine cell, depending upon where the M_3^- activity is inserted, thus creating a total of six cycles:

$\pi_{1,3} = \{M_3^+, M_1^-, M_2^-, M_3^-, M_3^+\}$ $\pi_{2,3} = \{M_3^+, M_1^-, M_3^-, M_2^-, M_3^+\}$
$\pi_{3,3} = \{M_3^+, M_3^-, M_1^-, M_2^-, M_3^+\}$ $\pi_{4,3} = \{M_3^+, M_2^-, M_3^-, M_1^-, M_3^+\}$
$\pi_{5,3} = \{M_3^+, M_2^-, M_1^-, M_3^-, M_3^+\}$ $\pi_{6,3} = \{M_3^+, M_3^-, M_2^-, M_1^-, M_3^+\}$.

We briefly interpret two examples of these cycles. In cycle $\pi_{5,3}$, since M_2^- precedes M_1^-, machine M_1 must be occupied by a part when M_3^+ occurs; also, since M_3^- does not precedes M_2^-, machine M_2 must be free at that time. In cycle $\pi_{6,3}$, since M_3^- precedes M_2^-, machine M_2 must be occupied by a part when M_3^+ occurs; also, since M_2^- precedes M_1^-, machine M_1 must be occupied by a part at that time.

Each of the above three machine cycles generates four cycles in a four machine cell, depending upon where the M_4^- activity is inserted, thereby creating a total of 24 cycles, as shown below.

$\pi_{1,4} = \{M_4^+, M_1^-, M_2^-, M_3^-, M_4^-, M_4^+\}$, $\pi_{2,4} = \{M_4^+, M_1^-, M_2^-, M_4^-, M_3^-, M_4^+\}$,
$\pi_{3,4} = \{M_4^+, M_1^-, M_4^-, M_2^-, M_3^-, M_4^+\}$, $\pi_{4,4} = \{M_4^+, M_4^-, M_1^-, M_2^-, M_3^-, M_4^+\}$,
$\pi_{5,4} = \{M_4^+, M_1^-, M_3^-, M_2^-, M_4^-, M_4^+\}$, $\pi_{6,4} = \{M_4^+, M_1^-, M_3^-, M_4^-, M_2^-, M_4^+\}$,
$\pi_{7,4} = \{M_4^+, M_1^-, M_4^-, M_3^-, M_2^-, M_4^+\}$, $\pi_{8,4} = \{M_4^+, M_4^-, M_1^-, M_3^-, M_2^-, M_4^+\}$,
$\pi_{9,4} = \{M_4^+, M_3^-, M_1^-, M_2^-, M_4^-, M_4^+\}$, $\pi_{10,4} = \{M_4^+, M_3^-, M_1^-, M_4^-, M_2^-, M_4^+\}$,
$\pi_{11,4} = \{M_4^+, M_3^-, M_4^-, M_1^-, M_2^-, M_4^+\}$, $\pi_{12,4} = \{M_4^+, M_4^-, M_3^-, M_1^-, M_2^-, M_4^+\}$,
$\pi_{13,4} = \{M_4^+, M_2^-, M_3^-, M_1^-, M_4^-, M_4^+\}$, $\pi_{14,4} = \{M_4^+, M_2^-, M_3^-, M_4^-, M_1^-, M_4^+\}$,
$\pi_{15,4} = \{M_4^+, M_2^-, M_4^-, M_3^-, M_1^-, M_4^+\}$, $\pi_{16,4} = \{M_4^+, M_4^-, M_2^-, M_3^-, M_1^-, M_4^+\}$,
$\pi_{17,4} = \{M_4^+, M_2^-, M_1^-, M_3^-, M_4^-, M_4^+\}$, $\pi_{18,4} = \{M_4^+, M_2^-, M_1^-, M_4^-, M_3^-, M_4^+\}$,
$\pi_{19,4} = \{M_4^+, M_2^-, M_4^-, M_1^-, M_3^-, M_4^+\}$, $\pi_{20,4} = \{M_4^+, M_4^-, M_2^-, M_1^-, M_3^-, M_4^+\}$,
$\pi_{21,4} = \{M_4^+, M_3^-, M_2^-, M_1^-, M_4^-, M_4^+\}$, $\pi_{22,4} = \{M_4^+, M_3^-, M_2^-, M_4^-, M_1^-, M_4^+\}$,
$\pi_{23,4} = \{M_4^+, M_3^-, M_4^-, M_2^-, M_1^-, M_4^+\}$, $\pi_{24,4} = \{M_4^+, M_4^-, M_3^-, M_2^-, M_1^-, M_4^+\}$.

The second index in $\pi_{i,j}$ will be omitted when it is clear from the context.

A.1.2 1-Unit Cycles in Activity Notation

We now describe 1-unit cycles for $m = 2, 3$, and 4, using the more popular activity notation. Recall from Chapter 2 that activity A_i represents the following:

- The robot unloads a part from M_i.
- The robot travels from M_i to M_{i+1}.
- The robot loads this part onto M_{i+1}.

The sequence of actions $(M_2^- M_4^- M_5^-)$ is represented as $(A_1 A_3 A_4)$. Since a part must be processed on all m machines and then placed into the output buffer, one instance of each of the $m+1$ activities $A_0, A_1, \ldots, A_m$ is required to produce a part. Then,

$\pi_{1,2} = \{A_0, A_1, A_2\}$ $\quad$ $\pi_{2,2} = \{A_0, A_2, A_1\}$.

$\pi_{1,3} = \{A_0, A_1, A_2, A_3\}$ $\quad$ $\pi_{2,3} = \{A_0, A_2, A_1, A_3\}$
$\pi_{3,3} = \{A_0, A_1, A_3, A_2\}$ $\quad$ $\pi_{4,3} = \{A_0, A_3, A_1, A_2\}$
$\pi_{5,3} = \{A_0, A_2, A_3, A_1\}$ $\quad$ $\pi_{6,3} = \{A_0, A_3, A_2, A_1\}$.

$\pi_{1,4} = \{A_0, A_1, A_2, A_3, A_4\}$ $\quad$ $\pi_{2,4} = \{A_0, A_1, A_3, A_2, A_4\}$
$\pi_{3,4} = \{A_0, A_3, A_1, A_2, A_4\}$ $\quad$ $\pi_{4,4} = \{A_0, A_1, A_2, A_4, A_3\}$

$\pi_{5,4} = \{A_0, A_2, A_1, A_3, A_4\}$ $\pi_{6,4} = \{A_0, A_2, A_3, A_1, A_4\}$
$\pi_{7,4} = \{A_0, A_3, A_2, A_1, A_4\}$ $\pi_{8,4} = \{A_0, A_2, A_1, A_4, A_3\}$
$\pi_{9,4} = \{A_0, A_1, A_3, A_4, A_2\}$ $\pi_{10,4} = \{A_0, A_3, A_1, A_4, A_2\}$
$\pi_{11,4} = \{A_0, A_1, A_4, A_2, A_3\}$ $\pi_{12,4} = \{A_0, A_1, A_4, A_3, A_2\}$
$\pi_{13,4} = \{A_0, A_3, A_4, A_1, A_2\}$ $\pi_{14,4} = \{A_0, A_4, A_1, A_2, A_3\}$
$\pi_{15,4} = \{A_0, A_4, A_1, A_3, A_2\}$ $\pi_{16,4} = \{A_0, A_4, A_3, A_1, A_2\}$
$\pi_{17,4} = \{A_0, A_2, A_3, A_4, A_1\}$ $\pi_{18,4} = \{A_0, A_3, A_2, A_4, A_1\}$
$\pi_{19,4} = \{A_0, A_2, A_4, A_1, A_3\}$ $\pi_{20,4} = \{A_0, A_2, A_4, A_3, A_1\}$
$\pi_{21,4} = \{A_0, A_3, A_4, A_2, A_1\}$ $\pi_{22,4} = \{A_0, A_4, A_2, A_1, A_3\}$
$\pi_{23,4} = \{A_0, A_4, A_2, A_3, A_1\}$ $\pi_{24,4} = \{A_0, A_4, A_3, A_2, A_1\}$.

The second index in $\pi_{i,j}$ is typically omitted when it is clear from the context.

Appendix B

B.1 The Gilmore-Gomory Algorithm for the TSP

The Gilmore-Gomory algorithm [67] solves a special case of the traveling salesman problem (TSP) in polynomial time. In this special case, each city $i, i = 1, ..., n$, is associated with two numerical parameters e_i and f_i, and the cost (distance) of traveling from city i to city j is given by $h_{ij} = \max\{e_j, f_i\}$. The objective is to find a tour ψ (i.e., a permutation of the cities) that minimizes the total cost $\sum_{i=1}^{n} \max\left\{e_{\psi(i+1)}, f_{\psi(i)}\right\}$. The classical two-machine no-wait flowshop problem, denoted $F_2|no\text{-}wait|C_t$, can be formulated as a TSP with the same cost structure and can be solved in time $O(n \log n)$ by the Gilmore-Gomory [67] algorithm. Several $O(n \log n)$ implementations of this algorithm are available; see, for example, Lawler et al. [103] and Vairatarakis [152]. In this appendix we describe the steps of the Gilmore-Gomory algorithm and illustrate them with a simple example. We first present a brief introduction to Problem $F_2|no\text{-}wait|C_t$, as this problem is closely related to several robotic cell problems discussed in this book. One simple implementation of the Gilmore-Gomory algorithm is then illustrated.

B.1.1 The Two-Machine No-Wait Flowshop Problem

In no-wait flowshops, each job must be processed from start to finish without any interruption on or between the machines. In $F_2|no\text{-}wait|C_t$, job $J_j, j = 1, 2, ..., n$, in a minimal-part-set (MPS) is processed first on machine $M_1^{'}$ for e_j time units and then immediately processed on the

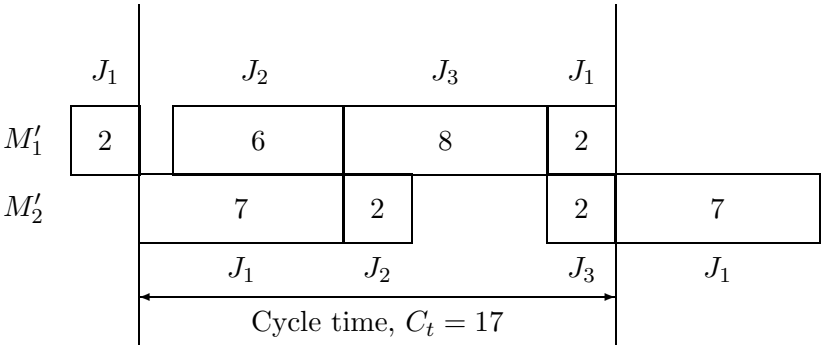


Figure B.1. A No-Wait Schedule in a Two-Machine Flowshop Corresponding to the Sequence $\psi = \{1, 2, 3\}$.

second machine $M_2^{'}$ for f_j time units. An MPS is processed repetitively every C_t units of time. The objective is to find a sequence ψ of the jobs that minimizes the cycle time C_t. To illustrate, consider an MPS of three jobs J_1, J_2, and J_3; their processing times on $M_1^{'}$ and $M_2^{'}$ are as follows: $e_1 = 2$, $f_1 = 7$; $e_2 = 6$, $f_2 = 2$; $e_3 = 8$, $f_3 = 2$. Figure B.1 shows a Gantt chart of the no-wait schedule that produces this MPS in the job sequence (J_1, J_2, J_3); that is, $\psi(i) = i, i = 1, 2, 3$. This schedule has a cycle time of 17. Because of the no-wait constraint, job J_2 cannot start immediately after the completion of J_1 on $M_1^{'}$.

B.1.2 Formulating a TSP

To formulate Problem $F_2|no\text{-}wait|C_t$ as a TSP, we define inter-city distances $h_{ij}, 1 \leq i, j \leq n, i \neq j$, as follows. Conceptually, we can let the next "cycle" begin and end with job J_1 (Figure B.2). The distance h_{ij} between cities i and j for the corresponding TSP is defined as the time between the start time of job J_i on $M_2^{'}$ and the start of job J_j on $M_2^{'}$. Thus,

$$h_{ij} = \max\{e_j, f_i\}.$$

The cycle time C_t may now be expressed as

$$C_t = \textstyle\sum_{i=1}^{n} h_{\psi(i),\psi(i+1)},$$

where we let $e_{\psi(n+1)} = e_{\psi(1)}$, since we need a cyclic schedule.

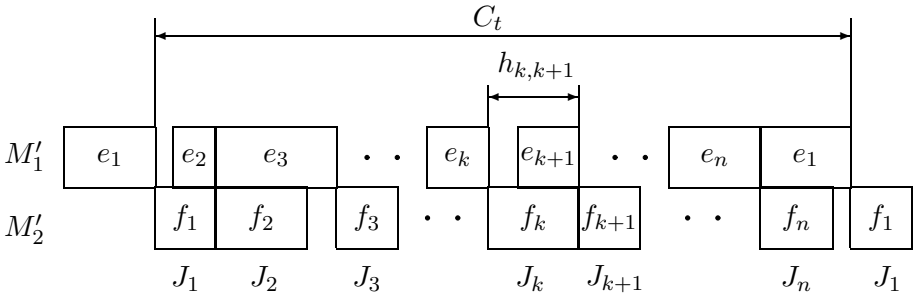


Figure B.2. Formulating $F_2|$*no-wait*$|C_t$ as a TSP.

Thus, by expressing the elapsed time between the start of two successive jobs J_i and J_j on the second machine $M_2^{'}$ as the distance between cities i to j, the no-wait problem $F_2|$*no-wait*$|C_t$ can be converted to a TSP (see Figure B.2). The Gilmore-Gomory algorithm solves the TSP under this special distance matrix in time $O(n \log n)$. The intuition behind the algorithm is that, ideally, the shortest processing time on the first machine should be concurrent with that on the second machine; similarly for the second shortest processing times on the two machines, and so on. If this is possible, we clearly have an optimal tour. If not, we have subtours. In the latter case, an optimal subtour-patching procedure moves the current schedule towards feasibility at minimum cost. We provide the algorithm in the next subsection; this procedure is also described in Kabadi [87] and Pinedo [132].

B.1.3 The Gilmore-Gomory Algorithm

Step 1: Sort $f_j, j = 1, 2, ..., n$, in non-decreasing order and re-number the jobs so that with the new numbering $f_j \leq f_{j+1}$, $j = 1, 2, ..., n-1$.

Step 2: For $p = 1, ..., n$, let $\phi(p)$ denote the index of the pth smallest element of the set $\{e_j; j = 1, ..., n\}$ in the new numbering of Step 1. It follows that $e_{\phi(j)} \leq e_{\phi(j+1)}$, $j = 1, 2, ..., n-1$.

Step 3: Compute the cost of arcs $(j, j+1), j = 1, 2, ..., n-1$ as follows:

$$c_{j,j+1} = \max\{0, \{\min(f_{j+1}, e_{\phi(j+1)}) - \max(f_j, e_{\phi(j)})\}\}.$$

Step 4: Construct an undirected graph with n nodes and arcs $(j, \phi(j))$, $j = 1, 2, ..., n$.

Step 5: If the current graph has only one component, go to Step 7. Otherwise, select the smallest value $c_{j,j+1}$ such that j is in one component and $j + 1$ in another. In the case of tie for smallest, choose arbitrarily.

Step 6: Add an undirected arc $(j, j + 1)$ to the graph, where j is the index selected in Step 5. Return to Step 5.

Step 7: Divide the arcs added in Step 6 into two groups. Arcs $(j, j+1)$ for which $f_j \leq e_{\phi(j)}$ are included in Group 1, while arcs with $f_j > e_{\phi(j)}$ are included in Group 2.

Step 8: Let there be r arcs in Group 1. Let $j_i, i = 1, ..., r$, be such that j_i is the ith largest index such that arc $(j_i, j_i + 1)$ is in Group 1.

Step 9: Let there be k arcs in Group 2. Let $t_i, i = 1, ..., k$, be such that t_i is the ith smallest index such that arc $(t_i, t_i + 1)$ is in Group 2.

Step 10: For an arc (p, q) in Group 1 or Group 2, let $\alpha_{p,q}$ be defined as follows: $\alpha_{p,q}(p) = q, \alpha_{p,q}(q) = p$ and $\alpha_{p,q}(j) = j$, if $j \neq p, q$. The minimal cycle time is obtained by following the jth job by the job $\psi^*(j) = \phi(v)$, where v, computed recursively, equals

$$\alpha_{j_1,j_1+1}(\alpha_{j_2,j_2+1} \ldots (\alpha_{j_r,j_r+1}(\alpha_{t_1,t_1+1}(\alpha_{t_2,t_2+1} \ldots (\alpha_{t_k,t_k+1}(j))...)))...).$$

EXAMPLE B.1 Consider the eight-job problem given in Table B.1; j is the job number; e_j and f_j are the processing times on machines $M_1^{'}$ and $M_2^{'}$, respectively.

j	e_j	f_j	j	e_j	f_j
1	2	10	5	8	6
2	12	9	6	17	22
3	16	23	7	4	12
4	1	5	8	20	12

Table B.1. Jobs and Their Processing Times.

- After sorting f_j's in non-decreasing order and re-numbering the jobs, we obtain Table B.2 (Step 1).

j	Renamed jobs	f_j	e_j	j	Renamed jobs	f_j	e_j
1	J_1	5	1	5	J_5	12	4
2	J_2	6	8	6	J_6	12	20
3	J_3	9	12	7	J_7	22	17
4	J_4	10	2	8	J_8	23	16

Table B.2. Jobs Sorted in Non-decreasing Order of $f_j, j = 1, ..., n$ (Step 1).

- Sorting the entries in column e_j of Table B.2 in non-decreasing order, we get column $e_{\phi(j)}$ in Table B.3. Column $\phi(j)$ gives the corresponding job number of e_j's in Table B.2. Next, we calculate the maximum and minimum of columns f_j and $e_{\phi(j)}$ for each j. Using these and the equation in Step 3 of the Gilmore-Gomory algorithm, we calculate the costs $c_{j,j+1}$ (Table B.3).

j	f_j	$e_{\phi(j)}$	$\phi(j)$	$\max\{f_j, e_{\phi(j)}\}$	$\min\{f_j, e_{\phi(j)}\}$	$c_{j,j+1}$
1	5	1	1	5	1	0
2	6	2	4	6	2	0
3	9	4	5	9	4	0
4	10	8	2	10	8	2
5	12	12	3	12	12	0
6	12	16	8	16	12	1
7	22	17	7	22	17	0
8	23	20	6	23	20	-

Table B.3. Computation of the Arc-Costs $c_{j,j+1}, j = 1, ..., n-1$ (Step 3).

- Next, we construct a undirected graph (Figure B.3) with nodes corresponding to job numbers $j = 1$ to 8. By referring to columns j and $\phi(j)$ of Table B.3, we draw the undirected edges j-$\phi(j)$ on the graph (Step 4). Thus, we obtain edges (2, 4), (3, 5), and (6, 8) in Figure B.3.

  The graph in Figure B.3 has five components. Node 1 is in the first component, nodes 2 and 4 are in the second component, nodes 3 and

5 are in the third component, nodes 6 and 8 are in in the fourth component, and node 7 is in the fifth component.

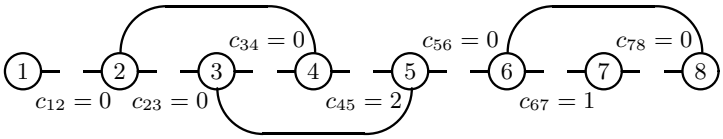


Figure B.3. Undirected Graph with Five Components (Step 4).

- Except edges (4, 5) and (6, 7), all other edges have a cost of zero. We, therefore, add undirected arcs (1, 2), (2, 3), (5, 6), and (7, 8) to the graph to obtain a single-component graph (Figure B.4). Steps 5 and 6 are now complete. Figure B.4 shows the final result.

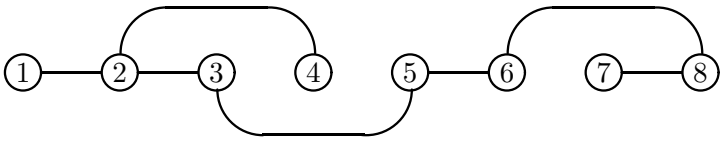


Figure B.4. A Single-Component Graph (Steps 5-6).

- An inspection of Table B.3 shows that the condition $f_j \leq e_{\phi(j)}$ is satisfied only by the arc (5, 6) (Step 7). Hence, from Step 7, we obtain two groups: Group 1 containing the arc (5, 6), and Group 2 containing arcs (1, 2), (2, 3), and (7, 8).

Group 1	Group 2
$f_j \leq e_{\phi(j)}$	$f_j > e_{\phi(j)}$
(5,6)	(1,2) (2,3) (7,8)

- Next, we execute Steps 8 and 9. The largest index j_1 such that arc $(j_1, j_1 + 1)$ is still in Group 1 is 5; so, $j_1 = 5$. The smallest index t_1 such that arc $(t_1, t_1 + 1)$ is in Group 2 is 1; thus, $t_1 = 1$. Similarly, $t_2 = 2$ and $t_3 = 7$.

- To obtain the index of the job following job j, we apply the permutation $\psi^*(j) = \phi(\alpha_{5,6}(\alpha_{1,2}(\alpha_{2,3}(\alpha_{7,8}(j)))))$. For example, for $j = 1$, $\psi^*(1) = \phi(\alpha_{5,6}(\alpha_{1,2}(\alpha_{2,3}(\alpha_{7,8}(1))))) = \phi(2) = 4$ (see Table B.3). Thus, job 4 follows job 1; the other computations can be done in a similar manner. An optimal job sequence (J_1, J_4, J_2, J_5, J_8, J_7, J_6, J_3) can be obtained from the pairs $(j, \psi^*(j))$ in Table B.4.

j	$\psi^*(j) = \phi(\alpha_{5,6}(\alpha_{1,2}(\alpha_{2,3}(\alpha_{7,8}(j)))))$
1	4
2	5
3	1
4	2
5	8
6	3
7	6
8	7

Table B.4. An Optimal Job Sequence.

- The Gantt chart in Figure B.5 gives the optimal makespan for this sequence. Note that the cycle time $C_t = 103$. For solving $F_2|no\text{-}wait|C_{max}$, we introduce an artificial job J_0 with $e_0 = 0$ and $f_0 = 0$, and re-run the above algorithm. In this case, we obtain the same job sequence with $C_{max} = 104$.

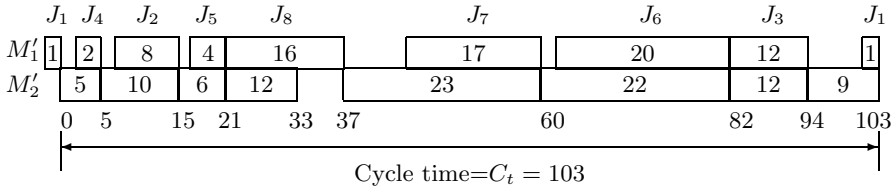


Figure B.5. A Gantt Chart for the Optimal Sequence.

B.2 The Three-Machine No-Wait Flowshop Problem as a TSP

In $F_3|no\text{-}wait|C_t$, job $J_j, j = 1, 2, ..., n$, in a minimal-part-set (MPS) is processed first on machine $M_1^{'}$ for e_j time units, then immediately processed on the second machine $M_2^{'}$ for f_j time units and then immediately on the third machine $M_3^{'}$ for g_j time units. An MPS is processed repetitively every C_t units of time. The objective is find a sequence ψ of the jobs that minimizes the cycle time C_t. The schedule corresponding to $\psi(i) = i, i = 1, 2, ..., n$, is shown in Figure B.6.

To formulate this problem as a TSP, we define inter-city distances $h_{ij}, 1 \leq i, j \leq n, i \neq j$, as follows. The distance h_{ij} between cities i and j for the corresponding TSP is defined as the time between the start time of job J_i on $M_1^{'}$ and the start of job J_j on $M_1^{'}$. Thus,

$$h_{ij} = \max\{e_i, e_i + f_i - e_j, e_i + f_i + g_i - e_j - f_j\}.$$

The cycle time C_t may now be expressed as

$$C_t = \sum_{i=1}^{n} h_{\psi(i),\psi(i+1)},$$

where we let $e_{\psi(n+1)} = e_{\psi(1)}$ and $f_{\psi(n+1)} = f_{\psi(1)}$, since we need a cyclic schedule.

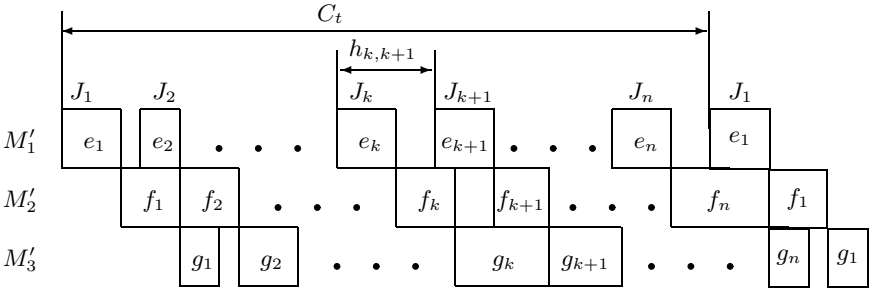


Figure B.6. Formulating $F_3|no\text{-}wait|C_t$ as a TSP.

References

[1] I.N.K. Abadi, N.G. Hall, C. Sriskandarajah, "Minimizing Cycletime in a Blocking Flowshop," *Operations Research*, Vol. 48, 177–180, 2000.

[2] A. Agnetis, "Scheduling No-Wait Robotic Cells with Two and Three Machines," *European Journal of Operational Research,* Vol. 123, 303–314, 2000.

[3] A. Agnetis, D. Pacciarelli, "Part Sequencing in Three Machine No-Wait Robotic Cells," *Operations Research Letters*, Vol. 27, 185–192, 2000.

[4] A.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows,* Prentice-Hall, Englewood Cliffs, NJ, 1993.

[5] R. Armstrong, L. Lei, S. Gu "A Bounding Scheme for Deriving the Minimal Cycle Time of a Single Transporter N-stage Process with Time Windows," *European Journal of Operational Research,* Vol. 78, 130–140, 1994.

[6] E. Akçalı, K. Nemoto, R. Uzsoy, "Cycle-Time Improvements for Photolithography Process in Semiconductor Manufacturing," *IEEE Transactions on Semiconductor Manufacturing,* Vol. 14, 48–56, 2001.

[7] Y.P. Aneja, H. Kamoun, "Scheduling of Parts and Robot Activities in a Two-Machine Robotic Cell," *Computers and Operations Research*, Vol. 26, 297–312, 1999.

[8] C.R. Asfahl, *Robots and Manufacturing Automation*, John Wiley & Sons, New York, 1985.

[9] R.G. Askin, C.R. Standridge. *Modeling and Analysis of Manufacturing Systems*, John Wiley & Sons, New York, 1993.

[10] W. Baumann, R. Birner, J. Haensler, R.P. Hartmann, A.B. Stevens, "Operating and Idle Times for Cyclic Multi-Machine Servicing," *The Industrial Robot,* 44–49, March, 1981.

[11] R. Bedini, G.G. Lisini, P. Sterpos, "Optimal Programming of Working Cycles for Industrial Robots," *Journal of Mechanical Design. Transactions of the ASME*, Vol. 101, 250–257, 1979.

[12] J.T. Black, "Cellular Manufacturing Systems Reduce Setup Time, Make Small Lot Production Economical," *Industrial Engineering*, 36-48, November, 1983.

[13] J.T. Black, B.J. Schroer, "Simulation of an Apparel Assembly Cell with Walking Workers and Decouplers," *Journal of Manufacturing Systems*, Vol. 12, 170-180, 1993.

[14] J. Błażewicz, H. Eiselt, G. Finke, G. LaPorte, J. Weglarz. "Scheduling Tasks and Vehicles in a Flexible Manufacturing System," *International Journal of Flexible Manufacturing Systems,* Vol. 4, 5–16, 1991.

[15] J. Błażewicz, S.P. Sethi, C. Sriskandarajah, "Scheduling of Robot Moves and Parts in a Robotic Cell," in *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications* (Ann Arbor, MI) K.E. Stecke and R. Suri (eds.), Elsevier, Amsterdam, 281–286, 1989.

[16] R.W. Bolz, *Manufacturing Automation Management*, Chapman and Hall, New York, 1985.

[17] N. Brauner, "Identical Part Production in Cyclic Robotic Cells: A State of the Art," Internal Note, Laboratoire Leibniz, Institut IMAG, Grenoble, France, 2006.

[18] N. Brauner, G. Finke, "Final Results on the One-Cycle Conjecture in Robotic Cells," Internal Note, Laboratoire Leibniz, Institut IMAG, Grenoble, France, 1997.

[19] N. Brauner, G. Finke, "Cyclic Scheduling in a Robotic Flowshop," in *Proceedings of the International Conference on Industrial Engineering and Production Management (IEPM)*, Vol. 1, 439–449, Lyon, France, 1997.

[20] N. Brauner, G. Finke, "On a Conjecture in Robotic Cells: New Simplified Proof for the Three-Machine Case," *INFOR,* Vol. 37, 20–36, 1999.

[21] N. Brauner, G. Finke, "Optimal Moves of the Material Handling System in a Robotic Cell," *International Journal of Production Economics*, Vol. 74, 269–277, 2001.

[22] N. Brauner, G. Finke, "Cycles and Permutations in Robotic Cells," *Mathematical and Computer Modeling,* Vol. 34, 565–591, 2001.

[23] N. Brauner, G. Finke, C. Gueguen, "Optimal Cyclic Production in Robotic Flowshops with Buffer," Techincal Report RR982-I, Laboratoire Leibniz, Institut IMAG, Grenoble, France, 1997.

[24] N. Brauner, G. Finke, W. Kubiak, "Complexity of One-Cycle Robotic Flow-Shops," *Journal of Scheduling*, Vol. 6, 355–371, 2003.

[25] P. Brucker, *Scheduling Algorithms*, Springer, Berlin, 1998.

[26] J.L. Burbidge, *The Introduction of Group Technology*, Heinemann, London, 1975.

[27] J.L. Burbidge, *Production Flow Analysis*, Clarendon Press, Oxford, 1996.

[28] J. Carlier, Ph. Chrétienne, *"Problémes d'Ordonnancement: Modélisation, Complexité Algorithmes*, Etudes et Recherche en Informatique, Masson, Paris, 1988.

[29] E. Chan, D. Lau, *A Scheduling Algorithm Implementation in a Robotic Cell,* Undergraduate Thesis, Department of Industrial Engineering, University of Toronto, 1994.

[30] A. Che, C. Chu, F. Chu, "Multicyclic Hoist Scheduling with Constant Processing Times," *IEEE Transactions on Robotics and Automation,* Vol. 18, 69–80, 2002.

[31] A. Che, C. Chu, E. Levner, "A Polynomial Algorithm for 2-degree Cyclic Robot Scheduling," *European Journal of Operational Research* , Vol. 145, 31–44, 2003.

[32] H. Chen, C. Chu, J. Proth, "Cyclic Scheduling of a Hoist with Time Window Constraints," *Rapport de Recherche Nr. 2307,* INRIA, 1994.

[33] H. Chen, C. Chu, J. Proth, "Cyclic Scheduling with Time Window Constraints," *IEEE Transactions on Robotics and Automation,* Vol. 14, 144–152, 1998.

[34] C. Chu, "A Faster Polynomial Algorithm for 2-Cyclic Robotic Scheduling," *Journal of Scheduling,* Vol. 9, 453–468, 2006.

[35] B.H. Claybourne, "Scheduling Robots in Flexible Manufacturing Cells," *CME Automation,* Vol. 30, 36–40, 1983.

[36] G. Cohen, D. Dubois, J.P. Quadrat, M. Viot, "A Linear-System-Theoretic Review of Discrete-Event Processes and Its Use for Performance Evaluation in Manufacturing," *IEEE Trans. Automation Control,* Vol. AC 30, 210–220, 1985.

[37] R.W. Cottle, J. Pang, R.E. Stone *The Linear Complementarity Problem.* Academic Press, New York, 1992.

[38] Y. Crama, "Combinatorial Optimization Models for Production Scheduling in Automated Manufacturing Systems," *European Journal of Operational Research,* Vol. 99, 136–153, 1997.

[39] Y. Crama, V. Kats, J. van de Klundert, E. Levner, "Cyclic Scheduling in Robotic Flowshops," *Annals of Operations Research: Mathematics of Industrial Systems*, Vol. 96, 97–124, 2000.

[40] Y. Crama, J. van de Klundert, "Cyclic Scheduling of Identical Parts in a Robotic Cell," *Operations Research,* Vol. 6, 952–965, 1997.

[41] Y. Crama, J. van de Klundert, "Robotic Flowshop Scheduling is Strongly NP-Complete," *Ten Years LNMB*, W.K. Klein Haneveld et al. (eds.), CWI Tract, Amsterdam, 277–286, 1997.

[42] Y. Crama, J. van de Klundert, "Cyclic Scheduling in 3-Machine Robotic Flowshops," *Journal of Scheduling,* Vol. 2, 35–54, 1999.

[43] E.M. Dar-El, Y. Rubinovitch, "MUST – a Multiple Solutions Technique for Balancing Single Model Assembly," *Management Science*, Vol. 25, 1105–1114, 1979.

[44] L. Davis, (ed.), *Handbook of Genetic Algorithms*, van Nostrand Reinhold, New York, 1991.

[45] M. Dawande, N. Geismar, S. Sethi, "Dominance of Cyclic Solutions and Some Open Problems in Scheduling Bufferless Robotic Cells," *SIAM Review*, Vol. 47, 709–721, 2005.

[46] M. Dawande, N. Geismar, S. Sethi, C. Sriskandarajah, "Sequencing and Scheduling in Robotic Cells: Recent Developments," *Journal of Scheduling,* Vol. 8, 387–426, 2005.

[47] M. Dawande, C. Sriskandarajah, S. Sethi, "On Throughput Maximization in Constant Travel-Time Robotic Cells," *Manufacturing and Service Operations Management*, Vol. 4, 296–312, 2002.

[48] V. Devedzic, "A Knowledge-Based System for the Strategic Control Level of Robots in Flexible Manufacturing Cell," *International Journal of Flexible Manufacturing Systems*, Vol. 2, 263–287, 1990.

[49] C. Dixon, S.D. Hill, "Workcell Cycle-Time Analysis in a Flexible Manufacturing System," *Proceedings of the Pacific Conference in Manufacturing,* Sydney and Melbourne, Australia, Vol. 1, 182–189, 1990.

[50] I.G. Drobouchevitch, S. Sethi, C. Sriskandarajah, "Scheduling Dual Gripper Robotic Cell: 1-Unit Cycles," *European Journal of Operational Research*, Vol. 171, 598–631, 2006.

[51] I.G. Drobouchevitch, S. Sethi, J. Sidney, C. Sriskandarajah, "Scheduling Multiple Parts in Two-Machine Dual Gripper Robotic Cell: Heuristic Algorithm

and Performance Guarantee," *International Journal of Operations and Quantitative Management*, Vol. 10, 297–314, 2004.

[52] I. Duenyas, J.W. Fowler, L.W. Schruben, "Planning and Scheduling in Japanese Semiconductor Manufacturing," *Journal of Manufacturing Systems,* Vol. 13, 323–333, 1994.

[53] S. Dunkerley, M.J. Adams, "A General Robot Control System for Automatic Chemical Analysis," *Laboratory Automation and Information Management*, Vol. 33, 93–105, 1997.

[54] G. Finke, C. Gueguen, N. Brauner, "Robotic Cells with Buffer Space," *Proceedings of the ECCO IX Conference*, R. O'Connor and P. Magee (eds.), Dublin City University, 1996.

[55] J.W. Fowler, M.C. Fu, L.W. Schruben, S. Brown, F. Chance, S. Cunningham, C. Hilton, M. Janakiram, R. Stafford, J. Hutchby, "Operational Modeling & Simulation in Semiconductor Manufacturing," D.J. Medeiros, E.F. Watson, J.S. Carson, M.S. Manivannan (eds.), *Proceedings of the Winter Simulation Conference,* 1035–1040, 1998.

[56] T. Ganesharajah, N.G. Hall, C. Sriskandarajah, "Design and Operational Issues in AGV-Served Manufacturing Systems," *Annals of Operations Research*, Vol. 76, 109-154, 1998.

[57] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, San Francisco, 1979.

[58] M.R. Garey, D.S. Johnson, R. Sethi, "The Complexity of Flowshop and Jobshop Scheduling," *Mathematics of Operations Research,* Vol. 1, 117–129, 1976.

[59] H.N. Geismar, M. Dawande, C. Sriskandarajah, "Robotic Cells with Parallel Machines: Throughput Maximization in Constant Travel-Time Cells," *Journal of Scheduling*, Vol. 7, 375–395, 2004.

[60] H.N. Geismar, M. Dawande, C. Sriskandarajah, "Approximation Algorithms for k-Unit Cyclic Solutions in Robotic Cells," *European Journal of Operational Research*, Vol. 162, 291–309, 2005.

[61] H.N. Geismar, M. Dawande, C. Sriskandarajah, "Throughput Optimization in Constant Travel-Time Dual Gripper Robotic Cells with Parallel Machines," *Production and Operations Management*, Vol. 15, 311–328, 2006.

[62] H.N. Geismar, M. Dawande, C. Sriskandarajah, "A 10/7 Approximation Algorithm for an Optimum Cyclic Solution in Additive Travel-Time Cells," *IIE Transactions*, Vol. 39, 217–227, 2007.

[63] H.N. Geismar, S.P. Sethi, J.B. Sidney, C. Sriskandarajah, "A Note on Productivity Gains in Flexible Robotic Cells," *International Journal of Flexible Manufacturing Systems*, Vol. 17, 5–21, 2005.

[64] H.N. Geismar, C. Sriskandarajah, N. Ramanan, "Increasing Throughput for Robotic Cells with Parallel Machines and Multiple Robots," *IEEE Transactions on Automation Science & Engineering*, Vol. 1, 84–89, 2004.

[65] M. Gendreau, A. Hertz, G. Laporte, "New Insertion and Postoptimization Procedures for the Traveling Salesman Problem," *Operations Research*, Vol. 40, 1086–1094, 1992.

[66] A.L. Giacobbe, "Diskette Labeling and Packaging System Features Sophisticated Robot Handling," *Robotics Today*, 73–75, April 1984.

[67] P. Gilmore, R. Gomory, "Sequencing a One-State Variable Machine: A Solvable Case of the Traveling Salesman Problem," *Operations Research,* Vol. 12, 675–679, 1964.

[68] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[69] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan, "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey," *Annals of Discrete Mathematics,* Vol. 5, 287–326, 1979.

[70] M.P. Groover, *Automation, Production Systems, and Computer-Integrated Manufacturing*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[71] M.P. Groover, M. Weiss, R. Nagel, N. Odrey, *Industrial Robotics: Technology, Programming, and Applications*, McGraw-Hill, New York, 1986.

[72] M.P. Groover, E.W. Zimmers, *CAD/CAM: Computer Aided Design and Manufacturing*, Prentice Hall, Englewood Cliffs, NJ, 1984.

[73] G. Gutin, A. Punnen (Eds.), *The Traveling Salesman Problem and Its Variations,* Kluwer Academic Publishers, Dordrecht, 2002.

[74] N.G. Hall, "Operations Research Techniques for Robotic Systems," *Handbook of Industrial Robotics,* S.Y. Nof (ed.), 2nd Edition, John Wiley & Sons, 1999.

[75] N.G. Hall, H. Kamoun, C. Sriskandarajah, "Scheduling in Robotic Cells: Classification, Two and Three Machine Cells," *Operations Research,* Vol. 45, 421–439, 1997.

[76] N.G. Hall, H. Kamoun, C. Sriskandarajah, "Scheduling in Robotic Cells: Complexity and Steady State Analysis," *European Journal of Operational Research,* Vol. 109, 43–63, 1998.

[77] N.G. Hall, C. Sriskandarajah, "A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process," *Operations Research,* Vol. 44, 510–525, 1996.

[78] N.G. Hall, M. Posner, "Generating Experimental Data for Computational Testing with Machine Scheduling Applications," *Operations Research*, Vol. 49, 854–865, 2001.

[79] N.G. Hall, C. Potts, C. Sriskandarajah, "Parallel Machine Scheduling with a Common Server," *Discrete Applied Mathematics,* Vol. 102, 223–243, 2000.

[80] J. Hartley, *Robots at Work*, North-Holland, Amsterdam, 1983.

[81] M. Hartmann, J.B. Orlin, "Finding Minimum Cost to Time Ratio Cycles with Small Integer Transit Times," *Networks,* Vol. 23, 567-574, 1993.

[82] J. Heizer, B. Render, *Operations Management, Seventh Edition*, Prentice-Hall, Englewood Cliffs, NJ, 2004.

[83] J. Herrmann, N. Chandrasekaran, B. Conaghan, M. Nguyen, G. Rubloff, R. Shi, "Evaluating the Impact of Process Changes on Cluster Tool Performance," *IEEE Transactions on Semiconductor Manufacturing,* Vol. 13, 181–192, 2000.

[84] F.S. Hillier, R.W. Boling, "On the Optimal Allocation of Work in Symmetrically Unbalanced Production Line Systems with Variable Operation Times," *Management Science*, Vol. 25, 721-728, 1983.

[85] I. Ioachim, F. Soumis, "Schedule Efficiency in a Robotic Production Cell," *The International Journal of Flexible Manufacturing Systems,* Vol. 7, 5–26, 1995.

[86] S.M. Johnson, "Optimal Two- and Three-Stage Production Schedules with Setup Times Included," *Naval Research Logistics Quarterly,* Vol. 1, 61–67, 1954.

[87] S.N. Kabadi, "Polynomially Solvable Cases of the TSP," in *The Traveling Salesman Problem and Its Variations,* G. Gutin and A.P. Punnen (Eds), Kluwer, Boston, MA, 489-584, 2002.

[88] E. Kafrissen, M. Stephans, *Industrial Robots and Robotics*, Reston Publishing, Reston, VA, 1984.

[89] H. Kamoun, *Scheduling in Repetitive Manufacturing Systems: Complexity, Heuristic Algorithms and System Design,* Ph.D. Thesis, Graduate Department of Industrial Engineering, University of Toronto, 1994.

[90] H. Kamoun, N.G. Hall, C. Sriskandarajah, "Scheduling in Robotic Cells: Heuristics and Cell Design," *Operations Research,* Vol. 47, 821–835, 1999.

[91] R.M. Karp, "A Characterization of the Minimum Cycle Mean in a Digraph," *Discrete Mathematics,* Vol. 3, 37-45, 1981.

[92] R.M. Karp, J.B. Orlin, "Parametric Shortest Path Algorithms with an Application to Cyclic Staffing," *Discrete Applied Mathematics,* Vol. 3, 37–45, 1981.

[93] V. Kats, "An Exact Optimal Cyclic Scheduling Algorithm for Multioperator Service of a Production Line, Part 2," *Automation and Remote Control*, Vol. 42, 538–543, 1982.

[94] V. Kats, E. Levner, "Polynomial Algorithms for Scheduling of Robots," in *Intelligent Scheduling of Robots and FMS,* E. Levner (ed.), Center for Technonlogical Education, Holon, Israel, 191–209, 1996.

[95] V. Kats, E. Levner, "Minimizing the Number of Robots to Meet a Given Schedule," *Annals of Operations Research*, Vol. 69, 209–226, 1997.

[96] V. Kats, E. Levner, "Polynomial Algorithms for Cyclic Scheduling of Tasks on Parallel Processors," *Proceedings of the 16th IASTED International Conference on Applied Informatics*, Garmisch, Germany, 302–304, 1998.

[97] V. Kats, E. Levner, "Cycle Scheduling in a Robotic Production Line," *Journal of Scheduling,* Vol. 5, 23–41, 2002.

[98] V. Kats, E. Levner, L. Meyzin, "Multiple-Part Cyclic Hoist Scheduling Using a Sieve Method," *IEEE Transactions on Robotics and Automation*, Vol. 15, 704–713, 1999.

[99] H. Kise, T. Shioyama, T. Ibaraki, "Automated Two-Machine Flowshop Scheduling: A Solvable Case," *IIE Transactions*, Vol. 23, 10-16, 1991.

[100] A.S. Kondoleon, "Cycle Time Analysis of Robot Assembly Systems," *Proceedings of the Ninth Symposium on Industrial Robots,* 575–587, 1979.

[101] S. Kumar, *Analytical and Metaheuristic Solutions for Emerging Scheduling Problems in E-commerce and Robotics,* Ph.D Thesis, The University of Texas at Dallas, 2001.

[102] S. Kumar, N. Ramanan, C. Sriskandarajah, "Minimizing Cycle Time in Large Robotic Cells," *IIE Transactions*, Vol. 37, 123-136, 2005.

[103] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (eds.), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* John Wiley & Sons: Chichester, 1985.

[104] E.L. Lawler, J. K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, "Sequencing and Scheduling: Algorithms and Complexity," in *Handbook of Operations Research and Management Science*, Vol. 4, 445-552, 1993.

[105] C-Y. Lee, L. Lei, M. Pinedo, "Current Trents in Deterministic Scheduling," *Annals of Operations Research,* Vol. 70, 1–41, 1997.

[106] T.E. Lee, M.E. Posner, "Performance Measures and Schedules in Periodic Job Shops," *Operations Research,* Vol. 45, 72-91, 1997.

[107] L. Lei, "Determining the Optimal Starting Times in a Cyclic Schedule with a Given Route," *Computers and Operations Research,* Vol. 20, 807-816, 1993.

[108] L. Lei, T.J. Wang, "Determining Optimal Cyclic Hoist Schedules in a Single-Hoist Electroplating Line," *IIE Transactions,* Vol. 26, 25–33, 1994.

[109] J.M.Y. Leung, G. Zhang, X. Yang, R. Mak, K. Lam, "Optimal Cyclic Multi-Hoist Scheduling: A Mixed Integer Programming Approach," *Operations Research*, Vol. 52, 965–976, 2004.

[110] E. Levner, "Optimal Planning of Parts Machining on a Number of Machines," *Automation and Remote Control,* Vol. 12, 1972–1981, 1969.

[111] E. Levner, V. Kats, V. Levit, "An Improved Algorithm for Cyclic Flowshop Scheduling in a Robotic Cell," *European Journal of Operational Research,* Vol. 97, 500–508, 1997.

[112] E. Levner, V. Kats, C. Sriskandarajah, "A Geometric Algorithm for Scheduling of Robots," in *Proceedings of the Second International Workshop on Intelligent Scheduling of Robots and Flexible Manufacturing Systems* (Holon, Israel), CTEH Press, 101–112, 1996.

[113] J.D.C. Little, "A Proof for the Queuing Formula: $L = \lambda W$," *Operations Research,* Vol. 9, 383–387, 1961.

[114] J. Liu, J. Jiang, Z. Zhou, "Cyclic Scheduling of a Single Hoist in Extended Electroplating Lines: a Comprehensive Integer Programming Solution," *IIE Transactions*, Vol. 34, 905–914, 2002.

[115] E.M. Livshits, Z.N. Mikhailetsky, E.V. Chervyakov, "A Scheduling Problem in an Automated Flow Line with an Automated Operator," *Computational Mathematics and Computerized Systems*, Vol 5, 151-155, 1974 (in Russian).

[116] R. Logendran, C. Sriskandarajah, "Sequencing of Robot Activities and Parts in Two-Machine Robotic Cells," *International Journal of Production Research,* Vol. 34, 3447–3463, 1996.

[117] R. Maiette, "Flexible Machining Cell with Robots," *Handbook of Industrial Robotics*, S. Nof (ed.), John Wiley and Sons, New York, 862–866, 1985.

[118] O.Z. Maimon, S.Y. Nof, "Coordination of Robots Sharing Assembly Tasks," *Journal of Dynamic Systems Measurement and Control. Transactions of the ASME,* Vol. 107, 299–307, 1985.

[119] F. Mangione, N. Brauner, B. Penz, "Optimal Cycles for the Robotic Balanced No-Wait Flowshop," in *Proceedings of the International Conference of Industrial Engineering and Production Management (IEPM)*, Porto, Portugal, May 2003.

[120] H. Matsuo, J.S. Shang, R.S. Sullivan, "A Crane Scheduling Problem in a Computer Integrated Manufacturing Environment," *Management Science,* Vol. 37, 587–606, 1991.

[121] R.K. Miller, *Robots in Industry: Applications for the Electronics Industry,* SEAI Institute: New York, 1984.

[122] R.K. Miller, T.C. Walker, *FMS/CIM Systems Integration Handbook*, The Fairmont Press, Lilburn, GA, 1990.

[123] T. Murata, H. Ishibuchi. "Performance Evaluation of Genetic Algorithms for Flowshop Scheduling Problems," *Proceedings of the First IEEE Conference on Evolutionary Computation*, Vol. 2, 812–817, 1994.

[124] S. Nahmias, *Production and Operations Analysis, Third Edition*, Irwin/McGraw-Hill, 1997.

[125] G. Nemhauser, L. Wolsey, *Integer Programming and Combinatorial Optimization,* John Wiley & Sons, New York, 1988.

[126] S.Y. Nof, D. Hannah, "Operational Characteristics of Multi-Robot Systems with Cooperation," *International Journal of Production Research,* Vol. 27, 477–492, 1989.

[127] C.H. Papadimitriou, P.C. Kannelakis, "Flowshop Scheduling with Limited Temporary Storage," *Journal of the Association of Computing Machinery,* Vol. 27, 533-549, 1980.

[128] T. Perkinson, P. McLarty, R. Gyurcsik, R. Cavin, "Single-Wafer Cluster Tool Performance: An Analysis of Throughput," *IEEE Transactions on Semiconductor Manufacturing,* Vol. 7, 369–373, 1994.

[129] T. Perkinson, R. Gyurcsik, P. McLarty, "Single-Wafer Cluster Tool Performance: An Analysis of the Effects of Redundant Chambers and Revisitation Sequences on Throughput," *IEEE Transactions on Semiconductor Manufacturing,* Vol. 9, 384–400, 1996.

[130] L.W. Phillips, P.S. Unger, "Mathematical Programming Solution of Hoist Scheduling Problem," *AIIE Transactions*, Vol. 8, 219–225, 1976.

[131] M. Pinedo, "Minimizing the Expected Makespan in Stochastic Flowshops," *Operations Research,* Vol. 30, 148–162, 1982.

[132] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems.* Prentice Hall, Englewood Cliffs, NJ, 2002.

[133] N. Ramanan, Personal Communication, 2002, 2004.

[134] Y. Rochat, "A Genetic Approach for Solving a Schduling Problem in a Robotized Analytical System," in *Intelligent Scheduling of Robots and FMS,* Ed. E. Levner (Center for Technonlogical Education, Holon, Israel), 191–209, 1996.

[135] S.S. Reddi, C.V. Ramamoorthy, "On the Flowshop Sequencing Problem with No Wait in Process," *Operational Research Quarterly,* Vol. 23, 323–330, 1972.

[136] H. Röck, "The Three Machine No-Wait Flowshop Problem is NP-Complete," *Journal of the Association for Computing Machinery*, Vol. 31, 336-345, 1984.

[137] R. Roundy, "Cyclic Schedules for Job Shops with Identical Jobs," *Mathematics of Operations Research,* Vol. 17, 842-865, 1992.

[138] D.A. Rudge, "The Automation of Solution Phase Synthetic Chemistry using XP Zymate Laboratory Robotic Systems," *Laboratory Automation and Information Management,* Vol. 33, 81–86, 1997.

[139] S. Sahni, Y. Cho, "Complexity of Scheduling Jobs with No Wait in Process," *Mathematics of Operations Research,* Vol. 4, 448-457, 1979.

[140] F. Sassani, "A Simulation Study on Performance Improvement of Group Technology Cells," *International Journal of Production Reserach*, Vol. 28, 293–300, 1990.

[141] A. Sciaky, "An Integrated Laser Processing Robotic Cell," *Handbook of Industrial Robotics*, S. Nof (ed.), John Wiley and Sons, New York, 867–878, 1985.

[142] S.P. Sethi, C. Sriskandarajah, G. Sorger, J. Błażewicz, W. Kubiak, "Sequencing of Parts and Robot Moves in a Robotic Cell," *International Journal of Flexible Manufacturing Systems,* Vol. 4, 331–358, 1992.

[143] S.P. Sethi, J. Sidney, C. Sriskandarajah, "Scheduling in Dual Gripper Robotic Cells for Productivity Gains," *IEEE Transactions on Robotics and Automation,* Vol. 17, 324–341, 2001.

[144] G.W. Shapiro, H.W. Nuttle, "Hoist Scheduling for a PCB Electroplating Facility," *IIE Transactions*, Vol. 20, 157–167, 1998.

[145] W. Song, Z.B. Zabinsky, R.L. Storch, "An Algorithm for Scheduling a Chemical Processing Tank Line," *Production Planning and Control*, Vol. 4, 323-332, 1993.

[146] C. Sriskandarajah, I. G. Drobouchevitch, S. Sethi, R. Chandrasekaran, "Scheduling Multiple Parts in a Robotic Cell Served by a Dual Gripper Robot," *Operations Research*, Vol. 52, 65–82, 2004.

[147] C. Sriskandarajah, N.G. Hall, H. Kamoun, "Scheduling Large Robotic Cells without Buffers," *Annals of Operations Research,* Vol. 76, 287–321, 1998.

[148] W.J. Stevenson, *Production/Operations Management, Seventh Edition*, Irwin/McGraw-Hill, 1999.

[149] Q. Su, F. Chen, "Optimal Sequencing of Double-Gripper Gantry Robot Moves in Tightly-Coupled Serial Production Systems," *IEEE Transactions on Robotics and Automation,* Vol. 12, 22–30, 1996.

[150] R. Suri, "Quantitative Techniques for Robotic Systems Analysis," *Handbook of Industrial Robotics,* Vol. I, S.Y. Nof (ed.), John Wiley & Sons, 1985.

[151] J.A. Tompkins, J.A. White, *Facilities Planning*, John Wiley & Sons, New York, 1984.

[152] G.L. Vairaktarakis, "Simple Algorithms for Gilmore-Gomory's Traveling Salesman and Related Problems," *Journal of Scheduling*, Vol. 6, 499–520, 2003.

[153] J. Van de Klundert, *Scheduling Problems in Automated Manufacturing,* Faculty of Economics and Business Administration, University of Limburg, Maastricht, The Netherlands, Dissertation No. 96-35, 1996.

[154] S. Venkatesh, R. Davenport, P. Foxhoven, J. Nulman, "A Steady-State Throughput Analysis of Cluster Tools: Dual-Blade Versus Single-Blade Robots," *IEEE Transactions on Semiconductor Manufacturing,* Vol. 10, 418–424, 1997.

[155] H.M. Wagner, "An Integer Programming Model for Machine Scheduling," *Naval Research Logistics Quarterly,* Vol. 6, 131–140, 1959.

[156] W.E. Wilhelm, "Complexity of Sequencing Tasks in Assembly Cells Attended by One or Two Robots," *Naval Research Logistics,* Vol. 34, 721–738, 1987.

[157] D.A. Wismer, "Solution of Flowshop Scheduling Problem with No Intermediate Queues," *Operations Research,* Vol. 20, 689–697, 1972.

[158] R.J. Wittrock, "Scheduling Algorithms for Flexible Flow Lines," *IBM Journal of Research and Development,* Vol. 29, 401–412, 1985.

[159] S. Wood, "Simple Performance Models for Integrated Processing Tools," *IEEE Transactions on Semiconductor Manufacturing,* Vol. 9, 320–328, 1996.

Web Sites Referenced

[160] Applied Robotics, Inc., *http://www.arobotics.com/products*, 2006.

[161] Asia Pacific Metal Working, *http://www.equipment-news.com/magazines/2000/sep2000/art05.htm*, 2000.

[162] Glass on Web, *http://www.glassonweb.com/articles/article/13/*, 2006.

[163] Manufacturing Talk, *http://www.manufacturingtalk.com/news/bab/bab101.txt*, 2006.

[164] Manufacturing Talk, *http://www.manufacturingtalk.com/news/guy/guy123.txt*, 2006.

[165] Products Finishing Magazine, *http://www.pfonline.com/articles/080103.html*, 2006.

[166] Rapid Development Services, Inc., *http://roboticintegrator.com/*, 2006.

[167] N. Remich, "The Robotic Ballet," *Appliance Manufacturer, http://www.ammagazine.com*, June 2000.

[168] RoboDesign International, Inc., *http://www.robodesign.com/roboarrayer_options.htm*, 2002.

[169] Robot Workspace Technologies, Inc., *http://www.rwt.com/RWT_Content_Files/articles/RWT_ASept99ME.html*, 1999.

[170] Robotics Online, *http://www.roboticsonline.com*, 2006.

[171] Sawyer and Smith Corporation, *http://www.buffingandpolishing.com*, 2006.

[172] Smart Media Group Ltd., *http://www.robotics-technology.com/articlearchive/2001/112001.shtml*, 2001.

[173] Vulcan Publications, Inc., *http://www.ndx.com/article.asp?article_id=270&channel_id=6*, 2001.

[174] Zmation, Inc., *http://www.zmation.com/products.htm*, 2006.

Copyright Permissions

Index

Early Titles in the
INTERNATIONAL SERIES IN OPERATIONS RESEARCH & MANAGEMENT SCIENCE
Frederick S. Hillier, Series Editor, *Stanford University*

Saigal/ *A MODERN APPROACH TO LINEAR PROGRAMMING*
Nagurney/ *PROJECTED DYNAMICAL SYSTEMS & VARIATIONAL INEQUALITIES WITH APPLICATIONS*
Padberg & Rijal/ *LOCATION, SCHEDULING, DESIGN AND INTEGER PROGRAMMING*
Vanderbei/ *LINEAR PROGRAMMING*
Jaiswal/ *MILITARY OPERATIONS RESEARCH*
Gal & Greenberg/ *ADVANCES IN SENSITIVITY ANALYSIS & PARAMETRIC PROGRAMMING*
Prabhu/ *FOUNDATIONS OF QUEUEING THEORY*
Fang, Rajasekera & Tsao/ *ENTROPY OPTIMIZATION & MATHEMATICAL PROGRAMMING*
Yu/ *OR IN THE AIRLINE INDUSTRY*
Ho & Tang/ *PRODUCT VARIETY MANAGEMENT*
El-Taha & Stidham/ *SAMPLE-PATH ANALYSIS OF QUEUEING SYSTEMS*
Miettinen/ *NONLINEAR MULTIOBJECTIVE OPTIMIZATION*
Chao & Huntington/ *DESIGNING COMPETITIVE ELECTRICITY MARKETS*
Weglarz/ *PROJECT SCHEDULING: RECENT TRENDS & RESULTS*
Sahin & Polatoglu/ *QUALITY, WARRANTY AND PREVENTIVE MAINTENANCE*
Tavares/ *ADVANCES MODELS FOR PROJECT MANAGEMENT*
Tayur, Ganeshan & Magazine/ *QUANTITATIVE MODELS FOR SUPPLY CHAIN MANAGEMENT*
Weyant, J./ *ENERGY AND ENVIRONMENTAL POLICY MODELING*
Shanthikumar, J.G. & Sumita, U./ *APPLIED PROBABILITY AND STOCHASTIC PROCESSES*
Liu, B. & Esogbue, A.O./ *DECISION CRITERIA AND OPTIMAL INVENTORY PROCESSES*
Gal, T., Stewart, T.J., Hanne, T./ *MULTICRITERIA DECISION MAKING: Advances in MCDM Models, Algorithms, Theory, and Applications*
Fox, B.L./ *STRATEGIES FOR QUASI-MONTE CARLO*
Hall, R.W./ *HANDBOOK OF TRANSPORTATION SCIENCE*
Grassman, W.K./ *COMPUTATIONAL PROBABILITY*
Pomerol, J-C. & Barba-Romero, S./ *MULTICRITERION DECISION IN MANAGEMENT*
Axsäter, S./ *INVENTORY CONTROL*
Wolkowicz, H., Saigal, R., & Vandenberghe, L./ *HANDBOOK OF SEMI-DEFINITE PROGRAMMING: Theory, Algorithms, and Applications*
Hobbs, B.F. & Meier, P./ *ENERGY DECISIONS AND THE ENVIRONMENT: A Guide to the Use of Multicriteria Methods*
Dar-El, E./ *HUMAN LEARNING: From Learning Curves to Learning Organizations*
Armstrong, J.S./ *PRINCIPLES OF FORECASTING: A Handbook for Researchers and Practitioners*
Balsamo, S., Personé, V., & Onvural, R./ *ANALYSIS OF QUEUEING NETWORKS WITH BLOCKING*
Bouyssou, D. et al./ *EVALUATION AND DECISION MODELS: A Critical Perspective*
Hanne, T./ *INTELLIGENT STRATEGIES FOR META MULTIPLE CRITERIA DECISION MAKING*
Saaty, T. & Vargas, L./ *MODELS, METHODS, CONCEPTS and APPLICATIONS OF THE ANALYTIC HIERARCHY PROCESS*
Chatterjee, K. & Samuelson, W./ *GAME THEORY AND BUSINESS APPLICATIONS*
Hobbs, B. et al./ *THE NEXT GENERATION OF ELECTRIC POWER UNIT COMMITMENT MODELS*
Vanderbei, R.J./ *LINEAR PROGRAMMING: Foundations and Extensions, 2nd Ed.*
Kimms, A./ *MATHEMATICAL PROGRAMMING AND FINANCIAL OBJECTIVES FOR SCHEDULING PROJECTS*
Baptiste, P., Le Pape, C. & Nuijten, W./ *CONSTRAINT-BASED SCHEDULING*
Feinberg, E. & Shwartz, A./ *HANDBOOK OF MARKOV DECISION PROCESSES: Methods and Applications*
Ramík, J. & Vlach, M./ *GENERALIZED CONCAVITY IN FUZZY OPTIMIZATION AND DECISION ANALYSIS*

Early Titles in the
INTERNATIONAL SERIES IN OPERATIONS RESEARCH & MANAGEMENT SCIENCE
(Continued)

Song, J. & Yao, D./ *SUPPLY CHAIN STRUCTURES: Coordination, Information and Optimization*
Kozan, E. & Ohuchi, A./ *OPERATIONS RESEARCH/ MANAGEMENT SCIENCE AT WORK*
Bouyssou et al./ *AIDING DECISIONS WITH MULTIPLE CRITERIA: Essays in Honor of Bernard Roy*
Cox, Louis Anthony, Jr./ *RISK ANALYSIS: Foundations, Models and Methods*
Dror, M., L'Ecuyer, P. & Szidarovszky, F./ *MODELING UNCERTAINTY: An Examination of Stochastic Theory, Methods, and Applications*
Dokuchaev, N./ *DYNAMIC PORTFOLIO STRATEGIES: Quantitative Methods and Empirical Rules for Incomplete Information*
Sarker, R., Mohammadian, M. & Yao, X./ *EVOLUTIONARY OPTIMIZATION*
Demeulemeester, R. & Herroelen, W./ *PROJECT SCHEDULING: A Research Handbook*
Gazis, D.C./ *TRAFFIC THEORY*
Zhu/ *QUANTITATIVE MODELS FOR PERFORMANCE EVALUATION AND BENCHMARKING*
Ehrgott & Gandibleux/ *MULTIPLE CRITERIA OPTIMIZATION: State of the Art Annotated Bibliographical Surveys*
Bienstock/ *Potential Function Methods for Approx. Solving Linear Programming Problems*
Matsatsinis & Siskos/ *INTELLIGENT SUPPORT SYSTEMS FOR MARKETING DECISIONS*
Alpern & Gal/ *THE THEORY OF SEARCH GAMES AND RENDEZVOUS*
Hall/ *HANDBOOK OF TRANSPORTATION SCIENCE - 2nd Ed.*
Glover & Kochenberger/ *HANDBOOK OF METAHEURISTICS*
Graves & Ringuest/ *MODELS AND METHODS FOR PROJECT SELECTION: Concepts from Management Science, Finance and Information Technology*
Hassin & Haviv/ *TO QUEUE OR NOT TO QUEUE: Equilibrium Behavior in Queueing Systems*
Gershwin et al./ *ANALYSIS & MODELING OF MANUFACTURING SYSTEMS*
Maros/ *COMPUTATIONAL TECHNIQUES OF THE SIMPLEX METHOD*
Harrison, Lee & Neale/ *THE PRACTICE OF SUPPLY CHAIN MANAGEMENT: Where Theory and Application Converge*
Shanthikumar, Yao & Zijm/ *STOCHASTIC MODELING AND OPTIMIZATION OF MANUFACTURING SYSTEMS AND SUPPLY CHAINS*
Nabrzyski, Schopf & Węglarz/ *GRID RESOURCE MANAGEMENT: State of the Art and Future Trends*
Thissen & Herder/ *CRITICAL INFRASTRUCTURES: State of the Art in Research and Application*
Carlsson, Fedrizzi, & Fullér/ *FUZZY LOGIC IN MANAGEMENT*
Soyer, Mazzuchi & Singpurwalla/ *MATHEMATICAL RELIABILITY: An Expository Perspective*
Chakravarty & Eliashberg/ *MANAGING BUSINESS INTERFACES: Marketing, Engineering, and Manufacturing Perspectives*
Talluri & van Ryzin/ *THE THEORY AND PRACTICE OF REVENUE MANAGEMENT*
Kavadias & Loch/ *PROJECT SELECTION UNDER UNCERTAINTY: Dynamically Allocating Resources to Maximize Value*
Brandeau, Sainfort & Pierskalla/ *OPERATIONS RESEARCH AND HEALTH CARE: A Handbook of Methods and Applications*
Cooper, Seiford & Zhu/ *HANDBOOK OF DATA ENVELOPMENT ANALYSIS: Models and Methods*
Luenberger/ *LINEAR AND NONLINEAR PROGRAMMING, 2nd Ed.*
Sherbrooke/ *OPTIMAL INVENTORY MODELING OF SYSTEMS: Multi-Echelon Techniques, Second Edition*
Chu, Leung, Hui & Cheung/ *4th PARTY CYBER LOGISTICS FOR AIR CARGO*
Simchi-Levi, Wu & Shen/ *HANDBOOK OF QUANTITATIVE SUPPLY CHAIN ANALYSIS: Modeling in the E-Business Era*

**** A list of the more recent publications in the series is at the front of the book ****