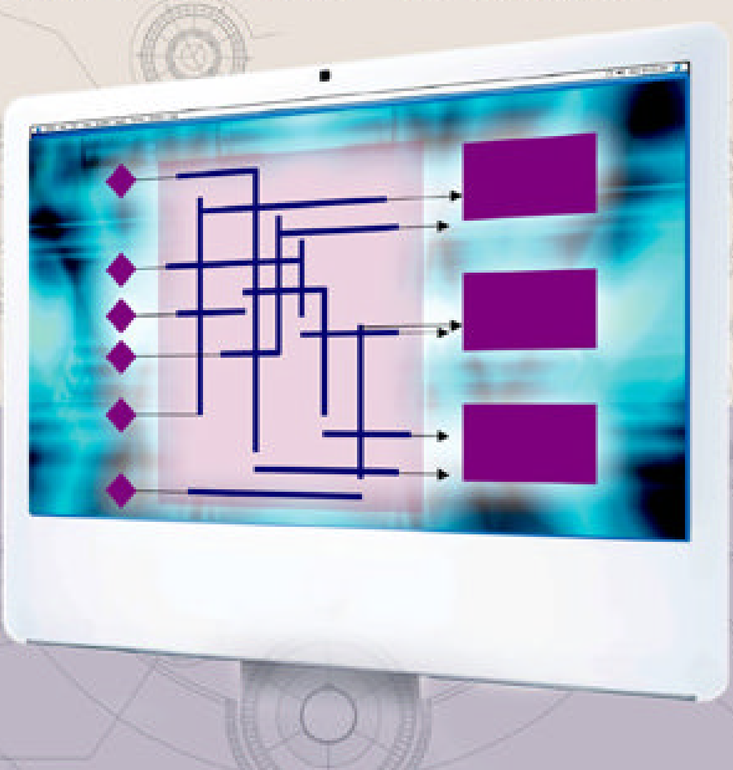
System Engineering and Operations Research

Vol. 1

STOCHASTIC SIMULATION OPTIMIZATION

An Optimal Computing Budget Allocation

Chun-Hung Chen • Loo Hay Lee



World Scientific

STOCHASTIC SIMULATION OPTIMIZATION

An Optimal Computing Budget Allocation

System Engineering and Operations Research — Vol. 1

STOCHASTIC SIMULATION OPTIMIZATION

An Optimal Computing Budget Allocation

Chun-Hung Chen

George Mason Univ., USA
National Taiwan Univ.

Loo Hay Lee

National Univ. of Singapore

World Scientific

NEW JERSEY • LONDON • SINGAPORE • BEIJING • SHANGHAI • HONG KONG • TAIPEI • CHENNAI

Published by

World Scientific Publishing Co. Pte. Ltd.

5 Toh Tuck Link, Singapore 596224

USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

System Engineering and Operations Research — Vol. 1
STOCHASTIC SIMULATION OPTIMIZATION
An Optimal Computing Budget Allocation

ISBN-13 978-981-4282-64-2
ISBN-10 981-4282-64-2

Typeset by Stallion Press
Email: enquiries@stallionpress.com

Printed in Singapore.

Foreword to the WSP Series on System Engineering and Operation Research

Advancement in science and technology often blurs traditional disciplinary boundary. Control system theory and practice, operations research, and computational intelligence combine to contribute to modern civilization in myriad ways. From traffic control on land, sea, and air, to manufacturing automation, to social and communication networks, these knowledge-based and human-made systems rely on research results in the above disciplinary topics for their smooth and efficient functioning.

The World Scientific Publishing Series on System Engineering and Operations Research is launched to fill this niche for students and scholars doing research in these areas. The first book in this series is by two leading scientists in the area of efficent simulation and modeling of complex systems. They articulate clearly the computational burden involved in such study and device innovative methodology to overcome the difficulties. I welcome their contribution in inaugurating this series and look forward to additional books in this genre.

Yu-Chi Ho
Editor-in-Chief
WSP Series on System Engineering
and Operations Research
January 20, 2010

This page intentionally left blank

Preface

"Simulation" and "optimization" are two very powerful tools in systems engineering and operations research. With the advance of new computing technology, simulation-based optimization is growing in popularity. However, computational efficiency is still a big concern because (i) in the optimization process, many alternative designs must be simulated; (ii) to obtain a sound statistical estimate, a large number of simulation runs (replications) is required for each design alternative. A user may be forced to compromise on simulation accuracy, modeling accuracy, and the optimality of the selected design. There have been several approaches developed to address such an efficiency issue. This book is intended to offer a different but ambitious approach by trying to answer the question "what is an optimal (or the most efficient) way to conduct all the simulations in order to find a good or optimal solution (design)?" The ultimate goal is to minimize the total simulation budget while achieving a desired optimality level, or to maximize the probability of finding the best design using a fixed computing budget. The primary idea is called Optimal Computing Budget Allocation (OCBA).

This book aims at providing academic researchers and industrial practitioners a comprehensive coverage of the OCBA approach for stochastic simulation optimization. Chapter 1 introduces stochastic simulation optimization and the associated issue of simulation efficiency. Chapter 2 gives an intuitive explanation of computing budget allocation and discusses its impact on optimization performance. Then a series of OCBA approaches developed for various problems are presented, from selecting the best design (Chapter 3), selecting a set of good enough designs (Chapter 5), to optimization with multiple objectives (Chapter 6). Chapter 4 provides numerical illustrations, showing that the computation time can be reduced significantly. Chapter 4 also offers guidelines for practical implementation of OCBA. Chapter 7 extends OCBA to large-scale simulation optimization problems. The OCBA technique is generic enough that it can be integrated with many optimization search algorithms to enhance simulation optimization efficiency. Several potential search techniques are explored. Finally, Chapter 8 gives a generalized view of the OCBA framework, and shows several examples of how the notion of OCBA can be extended to problems beyond simulation and/or optimization such as data envelopment analysis, experiments of design, and rare-event simulation. To help those readers without much simulation background, in the appendix, we offer a short but comprehensive presentation of stochastic simulation basics. We also include a basic version of OCBA source code in the appendix.

OCBA has several strengths: it is effective, easy to understand, simple to implement, and can be easily generalized or integrated with other methods to extend its applicability. We believe that this book is highly useful for different purposes.

1. For researchers, this book offers a series of promising approaches for efficiency enhancement in computer simulation, stochastic optimization, statistical sampling, and ranking and selection. The generalized framework may lead to numerous new lines of researches.
2. For courses, this book could serve as a textbook for advanced stochastic simulation or simulation optimization courses. They can cover Appendix A, Chapters 1 and 2 for introductions; Chapters 3 through 4, and parts of Chapters 5 through 8 for advanced materials.

3. For practioners, this book offers a simple but effective approach to enhance the computational efficiency of simulation optimization. Simulation practioners from industries, governments, and the military should find this book useful and relatively easy to read and apply because it gives numerous intuitive illustrations, well-structured algorithms, and practical implementation guidelines.

This page intentionally left blank

Acknowledgments

Many individuals and organizations have made the writing of this book possible. We are deeply grateful to our teachers for their mentorship, inspiration, and clarity, most especially Yu-Chi Ho (Harvard University), Peter B. Luh (University of Connecticut), and Shi-Chung Chang (National Taiwan University). The research in this book was supported directly and indirectly by National Science foundation, NASA Ames Research Center, National Science Council of Taiwan, George Mason University, National University of Singapore and National Taiwan University.

Our gratitude also extends to several graduate students who did most of the hard work. Jianwu Lin (Goldman Sachs) did the research in Chapter 3; Christopher Bluesteen (Systems Planning and Analysis, Inc.) is the author of Chapter 4, he ran the numerical experiments and wrote the chapter; Donghai He (CSSI) made significant contributions to Chapters 5 and 7. Suyan Teng (Singapore Republic Polytechnic), Juxin Li (National University of Singapore) and Si Zhang (National University of Singapore) has helped edit and conduct the numerical runs in Chapter 6. Mark Brantley (Jane's), Y. C. Luo (Chung-Cheng Institute of Technology), Ben Crain (George Mason University), and Wai-Peng Wong (University Science Malaysia) have contributions to Chapter 8.

We are deeply indebted to several long-time collaborators who have also been coauthors of publications related to the research presented in this book: John Birge (University of Chicago), Argon Chen (National Taiwan University), Ek Peng Chew (National University of Singapore), Stephen Chick (INSEAD), Liyi Dai (U.S. Army Research Office), Karen Donohue (University of Minnesota), David Goldsman (Georgia Institute of Technology), Douglas Morrice (University of Texas at Austin), Vicente Romero (Sandia National Laboratories), Leyuan Shi (University of Wisconsin and Beijing University), John Shortle (George Mason University), David Wu (Lehigh University), and Enlu Zhou (University of Illinois at Urbana-Champaign). We especially thank Enver Yücesan (INSEAD) and Michael Fu (University of Maryland), who have worked extensively with us on several OCBA publications and should be co-authors of this book.

We are also indebted to many colleagues and researchers who have influenced our thinking and offered support: Sigrun Andradottir (Georgia Institute of Technology), G. Anandalingam (University of Maryland), Russell Barton (Pennsylvania State University), Juergen Branke (University of Warwick), Xiren Cao (Hong Kong University of Science and Technology), Christos Cassandras (Boston University), Wai Kin (Victor) Chan (Rensselaer Polytechnic Institute), E. Jack Chen (BASF), Russell Cheng (University of Southampton), George Donohue (George Mason University), Peter Frazier (Cornell University), Peter Glynn (Stanford University), Wei-Bo Gong (University of Massachusetts), Monique Guignard-Spielberg (University of Pennsylvania), Patrick Harker (University of Delaware), Shane Henderson (Cornell University), Jeff Hong (Hong Kong University of Science and Technology), J. Q. Hu (Fudan University), Q. S. Jia (Tsinghua University), Sheldon Jacobson (University of Illinois), David Kelton (University of Cincinnati), Seong-Hee Kim (Georgia Institute of Technology), Michael E. Kuhl (Rochester Institute of Technology), Sujin Kim and Szu Hui Ng (National University of Singapore), Vijay Kumar (University of Pennsylvania), Marvin Nakayama (New Jersey Institute of Technology), Barry Nelson (Northwestern University), Lucy Pao (University of Colorado), Warren Powell (Princeton University), Bruce Schmeiser (Purdue University), Lee Schruben (University of California, Berkeley), Wheyming Tina Song (National Tsing Hua University), Roberto Szechtman (Naval Postgraduate School), Pirooz Vakili (Boston University), James Wilson (North Carolina State University), and Qian-Chuan Zhao (Tsinghua University).

We especially thank Ariela Sofer (Chair of the SEOR Department at George Mason University) for her encouragement and support at various stages in the process of writing this book. A big portion of the book was actually drafted in the department conference room next to her office. Special thanks also go to our editor, Gregory Lee, who has spent lots of effort to make the publication of this book possible.

Finally, we would like to thank our families (Mei-Mei, Flora, Valerie (who helped to draft the design of the book cover), Hannah and Jeremy) for their love, support, and patience during the writing of the book. Their contributions are just as real and important as the others named above, and we could not completed this project without them.

Thanks to all of you,
Chun-Hung Chen and Loo Hay Lee

This page intentionally left blank

Contents

Chapter 1

Introduction to Stochastic Simulation Optimization

1.1. Introduction

"Stochastic simulation optimization" (often shortened as simulation optimization) refers to stochastic optimization using simulation. Specifically, the underlying problem is stochastic and the goal is to find the values of controllable parameters (decision variables) to optimize some performance measures of interest, which are evaluated via stochastic simulation, such as discrete-event simulation or Monte Carlo simulation [Fu, 2002; Chen *et al.*, 2008; Fu *et al.*, 2008].

The act of obtaining the best decision under given circumstances is known as optimization. It is one of the leading topics of study and research. Optimization problems can be broadly classified into several branches depending upon the nature of the problem. For deterministic optimization problems, no randomness or uncertainty is considered and so the solutions have been relatively simple. The deterministic optimization approach might work well when the real problem has no noise or the uncertainty is not critical. However, many real-world optimization problems involve some sort of uncertainties in the form of randomness. These problems are studied under the branch "stochastic optimization" which plays a significant role in the design, analysis, and operation of modern systems. Methods for stochastic optimization provide a means of coping with inherent system noise. They may need to deal with models or systems that are highly nonlinear and high dimensional. The challenge is that these problems may become too complex to solve analytically due to complexity and stochastic relations. Attempts to use analytical models for such systems usually require many simplifying assumptions. As a matter of fact, the solutions are most likely to be inferior or inadequate for implementation. In such instances, a good alternative form of modeling and analysis available to the decision-maker is simulation.

Stochastic simulation is a powerful modeling and software tool for analyzing modern complex systems, because closed-form analytical solutions generally do not exist for such problems. Simulation allows one to accurately specify a system through the use of logically complex, and often non-algebraic, variables and constraints. Detailed dynamics of complex, stochastic systems can therefore be modeled. This capability complements the inherent limitations of traditional optimization. With such modeling advantages, simulation has been commonly used in so many different application areas, from transportation to manufacturing, telecommunication, supply chain management, health care, and finance. The combination of optimization and (stochastic) simulation is perhaps the Holy Grail of operations research/management science (OR/MS); see Fu [2007].

Examples of stochastic simulation optimization in health care include capacity planning of hospital beds and scheduling of surgical rooms/facilities. Pharmaceutical companies wish to choose the dosage level for a drug with the median aggregate response in patients. This dosage level is desirable because it achieves the positive effect of the drug while minimizing side effects. Similarly, one might wish to find the dosage level maximizing some utility function which is increasing in positive drug response and decreasing in negative drug response. Transportation systems that are commonly simulated include airspace, airports (gates, runways, baggage handling, passengers, taxi ways), rail networks, and roadway networks (urban, highway). In manufacturing, a factory simulation is commonly used for analyzing the operations of a semiconductor fabrication (fab) facility, as well as for supply chain networks. In network management, the decision-maker wants to select the fastest path through a network subject to traffic delays by sampling travel times through it. This network might be a data network through which we would like to transmit packets of data, or a network of roads through which we would like to route vehicles. In finance, simulation is commonly used for pricing and hedging of complex "exotic" derivatives, especially those that involve multiple underlying assets and sources of stochasticity, e.g., volatility, interest rates, and defaults.

For illustrative purpose, we give two simple examples as follows. Clearly, these examples could easily be extended to more realistic systems that include many items and complicated relationships.

Example 1.1 (Worker Allocation Problem). As shown in Figure 1.1, this two-node tandem queueing system includes two stages of service. Suppose there is a total of 31 workers who will be allocated to these two stages. Customers arrive at the first node and leave the system after finishing the services at both stages. The service at each stage is performed by one worker. At least one worker must be allocated to each stage. When multiple workers are available in one stage, the services for multiple customers are performed in parallel and independently, i.e., the service of one customer will not become faster even if there are more workers than customers in a stage. However, customers have to wait if all workers in that stage are all busy. Further, the workers assigned to one stage cannot help service at the other stage due to the training and safety requirement.

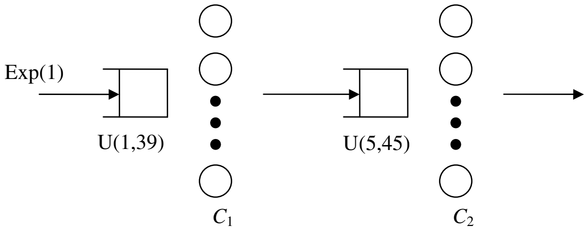


Figure 1.1. A two-node tandem queueing system with C_1 and C_2 workers at nodes 1 and 2, respectively.

The time to perform the service at stage 1 by one worker is uniformly distributed between 1 to 39 minutes, and the service time at stage 2 is uniformly distributed between 5 and 45 minutes. Customers arrive independently and the interarrival times between two customers are exponentially distributed with a rate of 1 customer per minute. To make customers happy, the manager of this service line wants the total time that a customer spends in the system to be as short as possible. A design question is how we should allocate these 11 workers so that the average total time in system (also called system time) is minimized.

Denote C_1 and C_2 as the numbers of workers allocated to nodes 1 and 2. Thus $C_1 + C_2 = 31$, $C_1 \geq 1$, and $C_2 \geq 1$. There are 30 alternative combinations of (C_1, C_2). We want to choose the best alternative of (C_1, C_2) so that the average system for the first 100 customers is minimized. Since there is no closed-form analytical solution for the estimation of the system time, stochastic simulation must be performed.

Consider another scenario where we allocate at least 11 workers to each stage, i.e., $C_1 \geq 11$, and $C_2 \geq 11$. Then the number of alternative combinations of (C_1, C_2) is only 10. However we still have the same question of finding the best design even though the number of alternatives is smaller. #

Example 1.2 ((s, S) Inventory Control Problem).
The second example is an (s, S) inventory policy problem based on the example given in Section 1.5.1 of Law and Kelton [2000]. Recall that under an (s, S) inventory control policy, when the inventory position (which includes inventory on hand and pipeline inventory) falls below s at an order decision point (discrete points in time in a periodic review setting and any point in time in a continuous review setting), then an order is placed in the amount that would bring the inventory position up to S.

In this example, the system involves a single item under periodic review, full backlogging, and random lead times (uniformly distributed between 0.5 and 1.0 period), with costs for ordering (including a fixed set-up cost of \$32 per order and an incremental cost of \$3 per item), on-hand inventory (\$1 per item per period), and backlogging (fixed shortage cost of \$5 per item per period). The times between demands are i.i.d. exponential random variables with a mean of 0.1 period. The sizes of demands are i.i.d. random variables taking values 1, 2, 3, and 4, with probability 1/6, 1/3, 1/3, and 1/6, respectively.

The usual performance measure of interest involves costs assessed for excess inventory, inventory shortages, and item ordering. Alternatively, the problem can be formulated with costs on excess inventory and item ordering, subject to a service level constraint involving inventory shortages. Even for a problem as simple as this one, there is no closed-form expression to illustrate the inventory cost. Stochastic simulation is performed to estimate the costs. A decision-maker wants to find the optimal values of (s, S) in order to minimize the inventory cost. #

In most cases, the predominant use of simulation is for performance evaluation, where common performance measures involve averages and probabilities of delays, cycle times, throughput, and cost. A typical simulation optimization framework consists of three levels of operations depicted in Figure 1.2. A simulation model is the

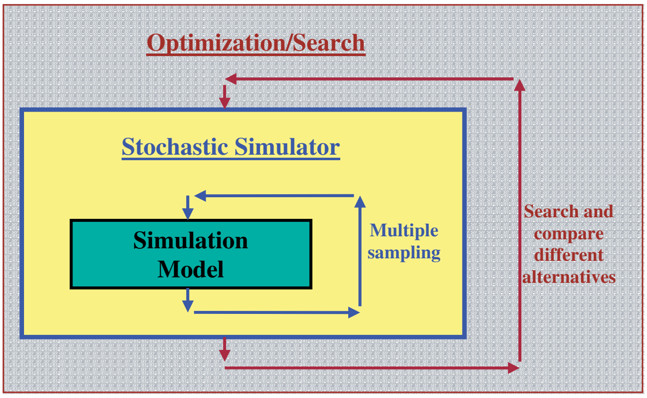


Figure 1.2. The structure of a typical simulation optimization process.

core of the framework and serves as the role of performance evaluator. For each alternative configuration of the decision variables, multiple simulation replications (or samples) must be performed in order to capture the property of randomness and obtain a statistical estimate. This constitutes the stochastic simulator which consists of a loop of multiple replications of the simulation model. The highest level is the optimization or search engine which may iteratively search the design space to find the best configuration of decision variables, where the stochastic simulator is applied to evaluate different candidate alternatives. These concepts will be further illustrated in the next section after we define our variables and problem.

While the advance of new technology has dramatically increased computational power, efficiency is still a big concern when using simulation for stochastic optimization problems. There are two important issues in dealing with the efficiency concern: i) at the level of stochastic simulator, a large number of simulation replications must be performed in order to capture randomness and obtain a sound statistical estimate at a specified level of confidence; and ii) at the level of optimization, many design alternatives must be well evaluated via stochastic simulation in order to determine the best design or to iteratively converge to the optimal design. Coupling these two issues together, the total computation cost grows very quickly. This becomes especially pronounced when the cost of simulation is not cheap. For example, one simulation replication of a complex semiconductor fab for a month of operations might take as long to run as solving a very large linear programming problem. There could easily be millions of random variates generated in the simulation, in which case one simulation replication can take minutes or hours to finish. A good evaluation of a single alternative design involving a large number of replications will then take hours or days to complete. If the number of design alternatives is large, the total simulation cost can be prohibitively expensive. A decision-maker is forced to compromise on simulation accuracy, modeling accuracy, and the optimality of the selected design. It is not surprising that in several industry applications, the stochastic nature of the problem is ignored or overlooked. With the presence of stochastic noise, a search technique may be misguided so that a system that seems "best" may not be the "true best". Hence an effective method to run the simulation efficiently is strongly desired.

1.2. Problem Definition

The setting is the general optimization problem which is to find a configuration, or design that minimizes the objective function:

$$\min_{\theta \in \Theta} J(\theta), \tag{1.1}$$

where θ is a p-dimensional vector of all the decision variables, commonly represented by x in mathematical programming, and Θ is the feasible region. If the objective function $J(\theta)$ is linear in θ and Θ can be expressed as a set of linear equations in θ, then we have a linear program, or mixed integer linear program if part of the Θ space involves an integer (e.g., $\{0, 1\}$ binary) constraint. Similarly, if $J(\theta)$ is convex in θ and Θ is a convex set, then we have a convex optimization problem. In general, Θ can be continuous, discrete, combinatorial, or even symbolic. Θ may be small or arbitrarily huge, well-structured or structureless.

If $J(\theta)$ can be evaluated in closed form and the solution can be found by solving analytically the necessary and/or sufficient conditions for the optimum, then no further discussions are needed. In the real world, unfortunately, many problems do not fall in this class. The setting in stochastic simulation optimization, however, presumes that we have little knowledge on the structure of J and moreover that J cannot be obtained directly, but rather is an expectation of another quantity $L(\theta, \omega)$, to which we have access, i.e.,

$$J(\theta) \equiv E[L(\theta, \omega)], \tag{1.2}$$

where ω comprises the randomness (or uncertainty) in the system and $L(\theta, \omega)$ is available only in the form of a complex calculation via simulation. The system constraints are implicitly involved in the simulation process, and so are not shown in Equation (1.2).

In our setting, a sample of ω represents a sample path or simulation replication, and $L(\theta, \omega)$ is a sample performance estimate obtained from the output of the simulation replication. For example,

$L(\theta, \omega)$ can be the number of customers who waited more than a certain amount of time in a queueing system, or average costs in an inventory control system, or the profit and loss distribution in an investment portfolio or risk management strategy. Most performance measures of interest can be put into this general form, including probabilities by using indicator functions and variances by estimating the second moments.

In the worker allocation problem given in Section 1.1, θ is one alternative of (C_1, C_2), and Θ is the set collecting all possible (C_1, C_2), i.e., $\{(C_1, C_2) | C_1 + C_2 = 11$, both C_1 and C_2 are integers and $\geq 1\}$. ω comprises the randomness including customer arrivals and services. $L(\theta, \omega)$ is an estimation of the average system time for the first 100 customers from one simulation run given a selection of (C_1, C_2). Our goal is to choose the best combination of (C_1, C_2), so that the expected average system time for the first 100 customers, i.e., $J(\theta)$, is minimized.

Similarly, in the inventory control problem, θ is one possible selection of (s, S) values, and Θ is the set collecting all possible (s, S). ω comprises the demand randomness including when the demands occur and their sizes. $L(\theta, \omega)$ is an estimation of the inventory cost during a designated period from one simulation run given a selection of (s, S). Our goal is to choose the best combination of (s, S), so that $J(\theta)$, the expected inventory cost, is minimized.

Multiple simulation replications must be performed in order to have a good estimate of $E[L(\theta, \omega)]$. Let N be the number of simulation samples (replications) and ω_j be the j-th sample of the randomness ω. Thus, $L(\theta, \omega_j)$ is the performance estimate obtained from the output of the simulation replication j. The standard approach is to estimate $E[L(\theta_i, \omega)]$ by the sample mean performance measure

$$\bar{J}(\theta) \equiv \frac{1}{N} \sum_{j=1}^{N} L(\theta, \omega_j). \tag{1.3}$$

As N increases, $\bar{J}(\theta)$ becomes a better estimate of $E[L(\theta, \omega)]$. Under some mild conditions, $\bar{J}(\theta) \to E[L(\theta, \omega)]$ as $N \to \infty$. With the notations and formulation, the stochastic simulation optimization

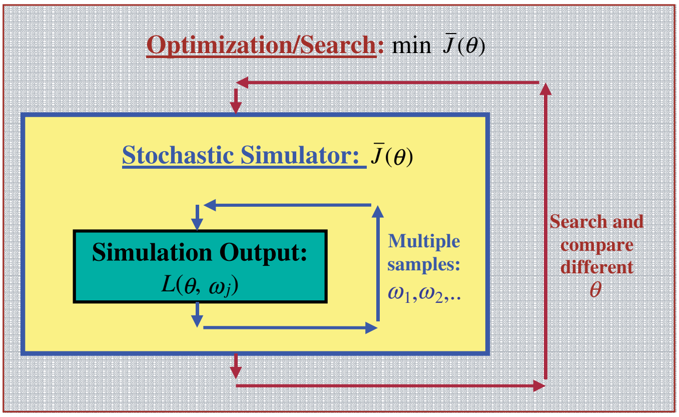

Optimization/Search: min $\bar{J}(\theta)$
Stochastic Simulator: $\bar{J}(\theta)$
Simulation Output: $L(\theta, \omega_j)$
Multiple samples: $\omega_1,\omega_2,..$
Search and compare different θ

Figure 1.3. The structure of a typical simulation optimization process.

framework previously given in Figure 1.2 becomes clearer as shown in Figure 1.3.

As discussed earlier, two primary efficiency concerns in stochastic simulation optimization are: i) N must be large if we want to have a sound estimate of $E[L(\theta, \omega)]$, which has an ultimate impact on the final solution that the optimization or search engine can find; and ii) $J(\theta)$ must be evaluated for many different θ in order to find the best θ.

It is a good idea to clarify some key terminologies used in the book. In the literature, there is a wide variety of terms used in referring to the inputs and outputs of a simulation optimization problem. Inputs are called (controllable) parameter settings, values, (decision) variables, (proposed) solutions, candidates, designs, alternatives, options, configurations, systems, or factors (in design of experiments terminology). Outputs are called performance measures, criteria, or responses (in design of experiments terminology). Some of the outputs are used to form an objective function, and there is a constraint set on the inputs. Following engineering design common usage, we will use the terms “design” and “objective function” in most of this book, with the latter comprised of performance measures estimated from simulation (consistent with discrete-event simulation common usage). A “design”, which is a particular setting of the variables (θ), can be interchangeably called an "alternative". Θ is called the search space or design space. In stochastic simulation, the replications of different random realizations can also be called samples, runs, or observations. Thus, if we have generated N samples of the randomness (i.e., $\omega_1, \omega_2, \ldots, \omega_N$) and conduct the corresponding stochastic simulations to obtain N simulation outputs (i.e., $L(\theta, \omega_1)$, $L(\theta, \omega_2), \ldots, L(\theta, \omega_N)$), then we may call them as N replications, N runs, or N samples.

1.3. Classification

Various methodologies have been developed for simulation optimization problems according to the sizes of design space (i.e., Θ). When the size of Θ is small, exhaustive enumeration is possible, but when the size is large, search algorithms need to be employed to explore the design space. These two types of problems are further discussed as follows.

1.3.1. *Design space is small*

The case of Θ being small implies that the design space is discrete and finite. In this case it is possible to simulate all candidate designs. Example 1.1 belongs to this category since it has 10 discrete designs. Example 1.2 can belong to this category if the decision-maker has only a small number of choices on the values of (s, S). Here is an example of 10 alternative inventory policies which are defined by the parameters $(s_1, s_2, \ldots, s_{10}) = (20, 20, 20, 40, 40, 40, 60, 60, 60, 80)$ and $(S_1, S_2, \ldots, S_{10}) = (30, 40, 50, 50, 60, 70, 70, 80, 90, 90)$, respectively.

In this category, enumeration can be used in principle to find the optimum. We can simply pick the best among a fixed set of designs, after all designs are simulated. However, unlike in deterministic optimization, "once" is not enough since the objective function estimate is noisy in the stochastic simulation setting. Hence the main question is how to conduct multiple simulation replications effectively for all designs in order to determine the optimum. This falls into the statistical ranking and selection literature (see, e.g., Bechhofer *et al.* [1995]).

In the context of simulation optimization, the difficulty of the search has been removed, so the focus is on efficiently allocating simulation replications among the alternative designs. This will be extensively discussed in Chapter 2. Among the simulation allocation approaches, Optimal Computing Budget Allocation (OCBA) schemes have been developed to significantly enhance the simulation efficiency by smartly allocating the computing budget among the compared designs [Chen, 1996; Chen *et al.*, 2000, 2008]. Further details will be presented in Chapters 3 and 4.

1.3.2. *Design space is large*

When Θ is large, enumeration becomes too expensive to conduct. In Example 1.2, suppose we have 1000 alternative options of s, and another 1000 alternative options of S. Then we have a total of one million alternative inventory policies for comparison in order to find the best design. Simulating all of the 1,000,000 alternative designs becomes infeasible. Some sorts of search like the ones in deterministic optimization must be applied to avoid simulating all the designs while ensuring a high chance of finding the best or a good design. Some approaches towards simulation optimization in this category include the following:

Model-based approaches. Implicitly we assume there is an underlining response function for $J(\theta)$. Iterative algorithms using statistical methods search the design space to improve upon the candidate design. There is also a non-sequential metamodel version (cf. Barton and Meckesheimer [2006]; Kleijnen [2008]). Some approaches utilize the gradient of the performance measure with respect to the parameters to help the search. The gradient-based approaches mimic gradient methods in deterministic (nonlinear) optimization to carry out local search. Unlike the deterministic counterpart, gradient estimation in stochastic simulation can be quite challenging due to the noisy output of stochastic simulation. There are two major types of methods for gradient estimation: i) One is the finite difference approach which estimates a derivative by simulating two different but very close design points. The difference of their objective function estimates is taken to estimate the derivative. One more efficient scheme is the simultaneous perturbation stochastic approximation introduced by Spall [1992], requiring only two points of simulations per gradient estimate, regardless of the dimension of the vector. The finite-difference approaches could be called "black box" methods, since no knowledge of the simulation model is used; ii) The second type of methods is the direct approaches. In the simulation setting, more is known about the underlying system, for example, distributions that generate input random variables. This allows for the implementation of more efficient direct methods to estimate the gradient. Methods for direct stochastic gradient estimation include perturbation analysis [Fu and Hu, 1997; Glasserman, 1991; Ho and Cao, 1991], the likelihood ratio/score function method [Rubinstein and Shapiro, 1993], and weak derivatives [Pflug, 1996]. For a more detailed overview on these methods, the reader is referred to Fu [2006].

Metaheuristics. Opposed to model-based approaches, this approach does not need the assumption of an underlining function, and is generally a gradient-free approach. This includes approaches such as genetic algorithms, evolutionary algorithms, simulated annealing, tabu search, scatter search, cross entropy, nested partition, particle swarm optimization, ant colony optimization, and other iterative and population-based algorithms from deterministic (nonlinear) optimization. Most of these metaheuristics start with an initial population of design(s). Then elite design(s) is(are) selected from this population of designs in order to generate a better population from iteration to iteration in the search process. The main differences between these approaches lie on how the new generation of designs are created, and how the elite designs are selected.

This book focuses on metaheuristics approaches, in which simulation plays a role of performance evaluation while the design space is explored and searched in the "optimization" phases as shown in Figures 1.2 and 1.3. Most search methods utilize the obtained information from the simulated designs to iteratively determine the search direction for the next step and hopefully will converge to the optimal design. The challenge is that at each iteration, many designs must be accurately simulated in order to obtain the guiding information for subsequent iterations, which can be very time consuming for stochastic problems. We will address this issue in Chapter 7.

1.4. Summary

In summary, the requirement of multiple replications for each design alternative is a critical concern for efficiency of stochastic simulation optimization. This concern becomes especially pronounced when the cost of simulation or sampling is not cheap. This book tries to answer the question "what is an optimal (or the most efficient) way to conduct all the simulations in order to find a good or optimal solution (design)?" The goal is to minimize the total simulation budget while achieving a desired optimality level.

This page intentionally left blank

Chapter 2

Computing Budget Allocation

As discussed in Chapter 1, the fact that a large number of simulation replications are often required to effectively distinguish between competing designs is a major challenge that often inhibits stochastic simulation optimization. The computation cost that accompanies the required simulation replications is expensive and often unacceptable. This book poses a potential solution to this challenge by employing analytic techniques to efficiently allocate the computing resources to each simulation experiment via the Optimal Computing Budget Allocation (OCBA) approach. The OCBA approach is complementary with the advance of new computational technology.

In this chapter, we introduce the premise of OCBA and give intuitive explanations of the most basic algorithms. We also motivate the need of variant allocation schemes under different scenarios and discuss its impacts if different computing budget allocation strategies are adopted.

For clarity, we will begin our examination by considering the case of a discrete and small design space. More general cases will be discussed at the end of this chapter.

2.1. Simulation Precision versus Computing Budget

Unlike deterministic optimization, "once" is not enough for stochastic simulation since the objective function estimate is noisy. As a result, multiple simulation replications are required in order to generate an accurate estimate of the objective function, $E[L(\theta, \omega)]$. As formulated in Section 1.2, $E[L(\theta, \omega)]$ is estimated by the sample mean performance measure

$$\bar{J}(\theta) \equiv \frac{1}{N} \sum_{j=1}^{N} L(\theta, \omega_j), \tag{2.1}$$

which is obtained via N simulation replications. Under some mild conditions, $\bar{J}(\theta) \to E[L(\theta, \omega)]$ as $N \to \infty$. It is impossible to generate an infinite number of simulation replications. However, as N increases, $\bar{J}(\theta)$ becomes a better estimate of $E[L(\theta, \omega)]$. The usual textbook approach (e.g., Law and Kelton [2000]; Banks *et al.* [2004]) for estimating the accuracy of a single performance measure is to utilize confidence intervals (e.g., a 95% confidence interval for mean system time may be 66 minutes ± 5 minutes). Another measure of precision is provided by the standard error

$$\frac{s}{\sqrt{N}},$$

where s is the sample standard deviation. Additionally, when the estimated performance is not heavily skewed, an approximation of the 95% confidence interval is constructed by taking two standard errors on both sides of the sample mean.

Based on the above methods, greater precision can be obtained by increasing the number of simulation replications, N. Roughly speaking, the precision of simulation output enhances at the rate of $\sqrt{N}$. However, by increasing N, the total computation cost will also increase.

There are various methods to counteract this increased computation cost. On one hand, one can buy a faster computer or optimize the simulation programming to expedite execution. Alternatively, one can apply efficient simulation algorithms. There exists a large amount of literature on innovative methods for improving simulation efficiency (see, e.g., Bratley *et al.* [1987]; Law and Kelton [2000] for a nice overview). Instead of performing the required N simulation replications faster, some of these approaches apply variance reduction techniques intended to reduce the simulation variance or standard deviation, s. As this variance is reduced, equal simulation precision can be obtained within less simulation replications. Among them, Control Variates (e.g., Nelson [1990]) attempt to take advantage of correlation between certain random variables to obtain the desired variance reduction. Antithetic Variates method induces negative correlation between pairs of simulation runs (e.g., Cheng [1982, 1984]; L'Ecuyer [1994]; Wilson [1983]) so that the variance of the sample mean estimator can be reduced. Glynn [1994] and Heidelberger [1993] deal with rare event problems by developing an importance sampling scheme. Another variance reduction approach for rare-event problems is splitting (e.g., Glasserman *et al.* [1999]; Garvels *et al.* [2002]; L'Ecuyer *et al.* [2006]). The basic idea of splitting is to create separate copies of the simulation whenever the simulation gets closer to the rare-event of interest. It has been shown that both splitting and importance sampling have potential to significantly reduce the variance for rare-event problems. It is worthy noting that the use of faster computers, optimized programming, or most variance reduction techniques are complementary with the development of the computing budget allocation methods presented in this book.

2.2. Computing Budget Allocation for Comparison of Multiple Designs

Traditionally, when using simulation to compare the performance of multiple designs, one gradually increases the computing budget (i.e., the number of simulation replications) for each alternative design until the standard error of the estimator is sufficiently small (i.e., the confidence interval for estimation is satisfactorily narrow) to select the most preferable design. One simple approach is to use an identical number of replications for each design, which can be inefficient, e.g., if one design has very low variance, then it may only require one or two simulation replications to estimate its performance.

Several approaches have been explored to improve the efficiency of selecting the best design. Approaches to address this problem fall under the well-established branch of statistics known as ranking and selection or multiple comparison procedures (see Bechhofer *et al.* [1995]). A useful notion defined in statistical ranking and selection procedures is the concept of *correct selection*, which refers to choosing a design (or configuration of the input variables) that optimizes the objective function. Correct selection can be defined as "correctly selecting the true best design", where the best is determined with respect to the smallest (or largest) mean.

Ranking and selection procedures introduce the concept of an *indifference zone* which provides a measure of closeness that the decision-maker tolerates away from absolute optimality. This is analogous to the *good enough* sets defined in ordinal optimization [Ho *et al.*, 2007]. In this case, correct selection corresponds to choosing a design whose objective function value is within the indifference zone of the optimal value. The usual statement of performance guarantees for these procedures is to specify a lower bound on the probability of correct selection. Dudewicz and Dalal [1975] developed a two-stage procedure for selecting a design that is within the indifference zone. In the first stage, all systems are simulated through some replications. Based on the results of the first stage, the number of additional simulation replications to be conducted for each design in the second stage is determined in order to reach the desired probability of correct selection. Rinott [1978] presents an alternative way to compute the required number of simulation replications in the second stage. Many researchers have extended this idea to more general ranking-and-selection settings in conjunction with new development (e.g., Bechhofer *et al.* [1995]; Matejcik and Nelson [1995]; Goldsman and Nelson [1998]). However, indifference-zone procedures determine the number of simulation replications based on a statistically conservative, least favorable configuration assumption. To some extent, least favorable configuration is a worst case condition when multiple designs are compared. Therefore, most well-known indifference-zone procedures are very conservative.

To improve the efficiency of ranking and selection, several approaches have been explored. Intuitively, to ensure a high probability of correct selection, a larger portion of the computing budget should be allocated to those designs that are critical to the process of identifying the best design. One of the approaches is the Optimal Computing Budget Allocation (OCBA) by Chen [1995] and Chen *et al.* [1997, 2000]. The approach relies upon the usage of both the sample means and variances in the allocation procedures, rather than assuming the least-favorable condition, and it intends to maximize the probability of correct selection. In the next section, we will further explain the ideas.

2.3. Intuitive Explanations of Optimal Computing Budget Allocation

As mentioned above, to ensure a high probability of correctly selecting an optimal design, a larger portion of the computing budget should be allocated to those designs that are critical in identifying the best design. In other words, a larger number of simulations must be conducted for critically competitive designs in order to reduce the estimation variance of these designs and maximize the probability of correct selection. On the other hand, limited computational effort should be expended on non-critical designs that have little chance of representing the best design (even if these designs have high variances). Based on this theory, the following questions remain: What are the "critical" designs? How should simulation time be allocated among critical and non-critical designs? This is a crucial question in the determination of efficient computing budget allocation. It turns out that the answer actually depends on what the current condition and our objective are.

We begin with the goal of maximizing the probability of correctly selecting the best design and explain the ideas using a series of simple examples. Suppose that in the inventory control problem (Example 1.2), five alternative values for (s, S) are provided to us. For each of these values we want to conduct simulation to find the one with the minimum expected cost. Suppose we conduct some initial simulation replications for all five alternative designs, and consider some possible outcomes from the first-stage estimates, in terms of the estimated inventory costs and associated 99% confidence intervals. The question of computing budget allocation is how we should allocate additional simulation replications if we want to continue more simulations to enhance the probability of correctly selecting the best design.

Case 2.1. A trivial example

Figure 2.1 gives a hypothetical example of possible results from the initial replications, showing the obtained 99% confidence intervals along with the accompanying mean estimator (represented as the line in the middle of the confidence interval). While there is uncertainty in the estimation of the performance for each design, it is obvious that designs 2 and 3 are much better than the other designs. As a result, it is sensible to allocate few or no simulation replications to designs 1, 4, and 5; instead allocating the bulk of the simulation budget to designs 2 and 3 provides more insight into the optimal solution. #

As discussed earlier, most widely used indifference-zone procedures assume least-favorable conditions, and as a result, they are extremely conservative and can be very inefficient. In addition, those procedures essentially allocate the simulation replications to each design proportional to the (estimated) variance, and do not consider the estimated means. In Case 2.1, the three worst designs have larger variance than the two best designs; as a result an allocation based solely on variances may waste resources on significantly inferior designs. In addition, they may not be able to distinguish which

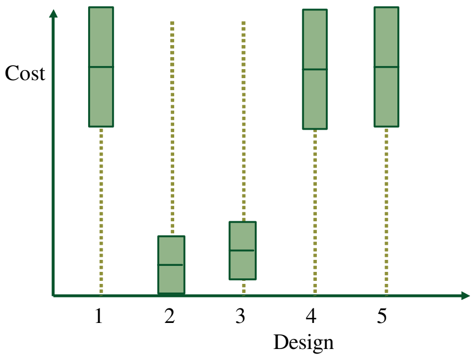


Figure 2.1. 99% confidence intervals for 5 alternative designs after preliminary simulation in Case 2.1.

of the two better designs is optimal. Thus, intuitively optimal simulation allocation would result in limited simulation of designs 1, 4, and 5, and more extensive simulation of designs 2 and 3.

One natural question that results from this "screening" approach is whether to always stop simulating poor performing designs or to simulate a very low number of replications of these designs. The answer is not necessarily always the same for all problem sets. This question is further explored in the following example.

Case 2.2. More simulation on inferior design

Again, we conduct some initial simulation replications for all five alternative designs. In this case, we assume the results produce the output shown in Figure 2.2. There is a significant amount of uncertainty in the performance estimation, but it appears that design 2 is better than designs 3, 4, and 5. Design 1 has the worst mean estimation but also has very large variance that overlaps with the performance of design 2.

For efficient simulation budget allocation, designs 3, 4, and 5 can be stopped because they are obviously inferior to design 2. Even though design 1 has the highest estimated inventory cost, it is a competitor with design 2 due to its high variability. To ensure design 2 is the correct selection, more simulation must be conducted on both designs 1 and 2. In this case, it is more effective to simulate design 1

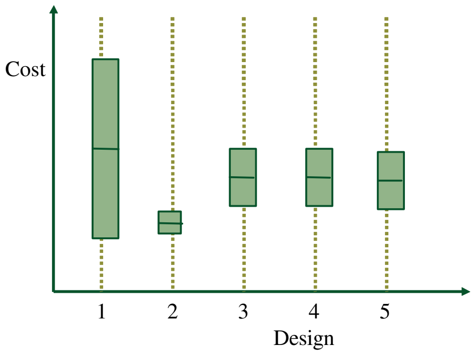


Figure 2.2. 99% confidence intervals for 5 alternative designs after preliminary simulation in Case 2.2.

because its variance is significantly higher than design 2; as a result, more simulation on design 1 will result in greater variance reduction and higher probability of correct selection. Thus, in this scenario, design 1 should receive most of the additional simulation, even though it is the worst design. #

Case 2.3. Most common scenario

In Cases 2.1 and 2.2, some designs dominate the others and so we have a good idea about how the additional simulation budget should be allocated. However, most cases are not as trivial as those shown above. We consider a more common scenario here.

Figure 2.3 shows the output after initial simulations. In this case, some designs seem better, but none are clearly better than all the others, since all the confidence intervals overlap. In situations such as this, it is not straightforward to determine which designs can be eliminated and which designs should receive more simulation budget. #

As shown in Case 2.3, a good computing budget allocation is not trivial. Ideally, we would like to allocate simulation replications to designs in a way that maximizes the simulation efficacy (i.e., the probability of correct selection). As discussed above, this approach is intuitively dependent upon the mean and variance of each design. Chapters 3 and 4 will extensively discuss the specific relationship

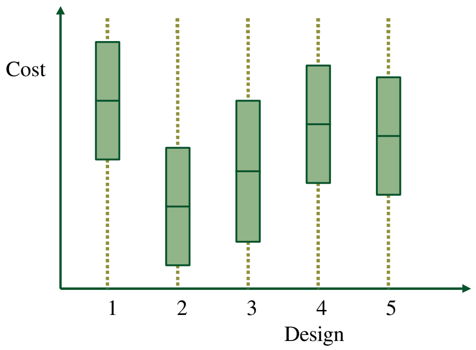


Figure 2.3. 99% confidence intervals for 5 alternative designs after preliminary simulation in Case 2.3.

between mean, variance, and the amount of simulation replications that should be optimally allocated to each design.

Moreover, a good computing budget allocation depends on not only the relative means and variances of all designs, but also the specific objective that we would like to achieve. In previous cases, our objective has been to choose the best design correctly. If we employ different objectives, the computing budget allocation may be entirely different. This issue is illustrated in the following two cases.

Case 2.4. Selecting a subset

In this example, instead of choosing the best design, we would like to choose all the top-m designs (i.e., to determine which designs are among the best top-m, where $m \geq 2$). Similarly, we need to conduct initial simulation replications for all five alternative designs. The results from these preliminary simulations are shown in Figure 2.4. In this case, the inventory cost estimations are in ascending order and all designs have similar variance.

Design 1 seems to be the best design and design 2 is a close competitor. All other designs have steadily decreasing performance. If our objective is to correctly select the best design, it is reasonable to stop simulating designs 3, 4, and 5, and only continue simulating designs 1 and 2.

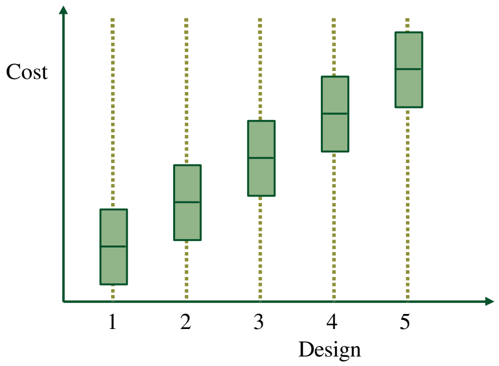


Figure 2.4. 99% confidence intervals for 5 alternative designs after preliminary simulation in Case 2.4.

However, if the decision-maker wants to identify the top-2 designs, instead of selecting the best design, how should we continue allocating simulation budget? Since design 1 is much better than designs 3, 4, and 5, it is clear that design 1 is probably one of the top-2 designs. As a result, there is minimal need to continue simulating design 1. On the other hand, designs 4 and 5 are much worse than the other designs and so they have little chance to be ranked within top-2. Therefore, there is not much value to simulate designs 4 and 5 either. To allocate computing budget efficiently, it is reasonable to concentrate simulation time on designs 2 and 3 because they are the "critical designs" in determining the top-2 designs.

Following this same idea, if we are interested in selecting the subset of top-3 designs, it is best to simulate most on designs 3 and 4. If we want to determine the worst design, an efficient simulation budget allocation would allocate most of the simulation budget to designs 4 and 5. #

The various scenarios in Case 2.4 demonstrate that the selection of "critical" designs for efficient simulation depends on the objective. More information on efficient computing budget allocation for selecting an optimal subset of top-m designs will be presented in Chapter 5.

Case 2.5 Multiple objectives

When there is more than one objective, how to select "best" designs will be different and so how the simulation budget should be allocated will be vastly different. Consider the same inventory problem (Example 1.2). Other than the inventory cost, we might also like to minimize the expected number of backlogged quantity. Similar to previous examples, suppose the results from preliminary simulation are shown in Figures 2.5 and 2.6. Figure 2.5 represents the estimation of inventory cost, while Figure 2.6 shows backlog quantity. Ideally, we would like to minimize both objectives.

First, we tentatively ignore the estimation variances and focus on the sample means. In terms of minimizing inventory cost, design 1 is the preferred option. However, design 2 has the minimum backlog quantity. Hence designs 1 and 2 should intuitively receive the most

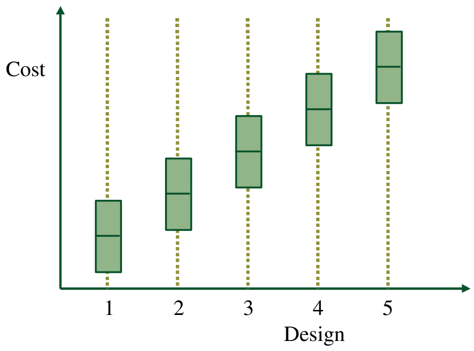


Figure 2.5. 99% confidence intervals (inventory cost) for 5 alternative designs after preliminary simulation in Case 2.5.

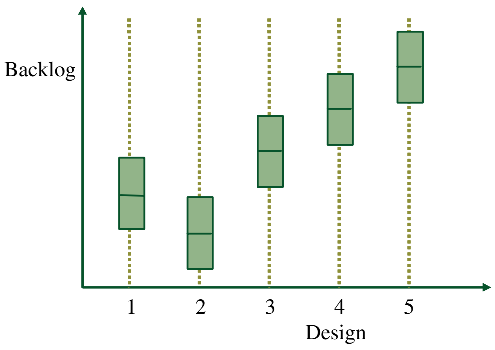


Figure 2.6. 99% confidence intervals (backlog quantity) for 5 alternative designs after preliminary simulation in Case 2.5.

additional simulation replications. Designs 3, 4, and 5 are worse than design 1 and 2 for both objectives; in other words, designs 3, 4, and 5 are dominated by designs 1 and 2. Designs 3, 4, and 5 are called dominated designs. In general, we aim at selecting all the non-dominated designs, which are probably designs 1 and 2 in this example.

When the estimation variance (i.e., simulation noise) is considered, design 3 may also require more simulation because it is a close competitor to designs 1 and 2 and its confidence interval overlaps with the confidence interval for design 2 on inventory cost and overlaps with the confidence interval for design 1 on the backlog quantity. #

Efficient computing budget allocation for multi-objective problem will be extensively discussed in Chapter 6.

2.4. Computing Budget Allocation for Large Simulation Optimization

As discussed in Section 1.3, when the design space Θ is large, it is too expensive to simulate all candidate designs. Some optimization search techniques must be used to explore and search the design space. In this setting, simulation plays the role of a performance evaluator. Most search methods utilize the information obtained from the simulation evaluation to iteratively determine the search direction for the next step and hopefully converge to an optimal or good design after some iterations. For stochastic simulation, such a search can be challenging due to simulation noise. We give an intuitive explanation using a simple local search method on a simple example which consists of a unique optimal design as shown in Figure 2.7.

Suppose we have randomly sampled two designs points: x_1 and x_2. After evaluating these two designs, we can determine the search direction to generate new designs for the next iteration. If the simulation is deterministic (i.e., there is no noise), this approach can identify a good search direction shown as the thick arrow in Figure 2.7.

However, when simulation is stochastic, the determination of search direction can be challenging. Figures 2.8 and 2.9 give two

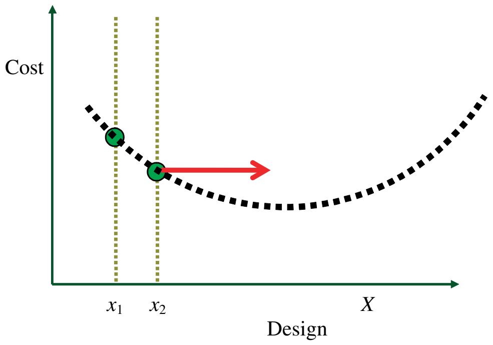


Figure 2.7. Determination of search direction for minimizing an objective function in a deterministic problem.

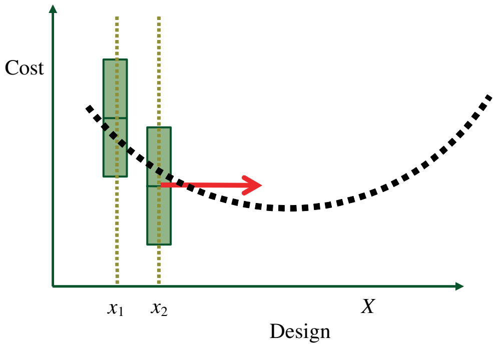


Figure 2.8. Determination of search direction when simulation noise is present.

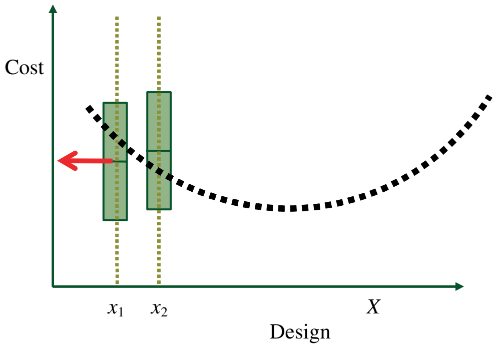


Figure 2.9. Determination of search direction when simulation noise is present.

hypothetical examples of possible results from the initial simulations, showing the obtained 99% confidence intervals along with the accompanying mean estimator. Due to the simulation noise, one may obtain the wrong search direction if the simulation precision is not high enough. The concern usually becomes even more serious when $J(x_1)$ and $J(x_2)$ are very close because very small noise can result in a wrong search direction. Thus one would need an even higher simulation precision in order to avoid a wrong search direction, which implies a very high simulation cost.

This is generally true for all of the search methods. On another example, genetic algorithms (e.g., GA, Holland [1975]; Goldberg [1989]) choose some good designs from a subset of candidate designs via simulation evaluation. These chosen good designs are then used to generate a new subset of designs for next iteration. By generating decedent designs from good parent designs, GA iteratively converges to an optimal design under some conditions. With simulation noise, if the simulation precision is not high, we may end up with selecting inferior designs as parent designs to make the next generation, which may misguide the direction of search or slow down the convergence rate.

From the computing budget allocation perspective, we aim to allocate the computing budget in a smarter way so that the search information can be correctly estimated in a most efficient manner. In the example of GA, we do not have to equally simulate all designs in the subset in order to find the good ones. Instead, we want to correctly choose the good designs using a minimum computing budget. This is similar with what we had explained in Case 2.4. Naturally, different search methods require different information and so the optimal computing budget allocation schemes are different. Extensive discussions will be given in Chapters 7 and 8.

2.5. Roadmap

Starting in the next chapter, we will present a series of optimal computing budget allocation (OCBA) methods designed for different problems. Chapter 3 lays foundations and presents a thorough derivation for OCBA on selecting the best design. Chapter 4 demonstrates its effectiveness using a series of numerical examples. Some advices about numerical implementation of the OCBA approach are offered. Instead of selecting the best design, Chapter 5 presents an OCBA algorithm for selecting the optimal subset of top-m designs. Chapter 6 deals with the computing budget allocation when a decision-maker is facing multiple objectives. Chapter 7 presents a framework on how to integrate the search methods (metaheuristics) with the OCBA method and shows some examples of such integration. Chapter 8 gives a generalized view of OCBA, and shows several examples on how the notion of OCBA can be extended to problems beyond simulation and/or optimization.

Chapter 3

Selecting the Best from a Set of Alternative Designs

Starting from this chapter, we will develop effective methodologies for Optimal Computing Budget Allocation (OCBA) in stochastic simulation optimization. We start with a simple case, where Θ is finite and small enough so enumeration can be used in principle to find the optimum. Specifically,

$$\Theta \equiv \{\theta_i, i = 1, 2, \ldots, k\},$$

where k is relatively small (e.g., $5 \sim 100$) as compared with other cases we will consider in later chapters. The alternative design with particular setting θ_i is called design i. $L(\theta_i, \omega)$ is a sample performance estimate obtained from the output of one simulation replication for design i. After performing N_i simulation replications for design i, $J_i \equiv J(\theta_i) = E[L(\theta_i, \omega)]$ is estimated using its sample mean

$$\bar{J}_i \equiv \frac{1}{N_i} \sum_{j=1}^{N_i} L(\theta_i, \omega_{ij}),$$

where ω_{ij} is the j-th simulation sample of ω for design i.

Section 3.1 presents a Bayesian framework which forms the basis of the OCBA methodology development. The rationale for the adoption of the Bayesian model is the ease of derivation of the solution approach. While the Bayesian model has some advantages in offering intuitive explanations of the methodology development and resulting allocation, the classical (frequentist) model works equally well in terms of developing the OCBA schemes. For most of this chapter, we will develop OCBA schemes under the Bayesian setting. At the end of this chapter (Section 3.7), we will consider one OCBA problem and show that the classical model works equally well and actually yields the same solution. Further details about comparison of the Bayesian model with the classical model for the simulation output modeling can be found elsewhere (e.g., Bernardo and Smith [1984]; Inoue and Chick [1998]; Inoue *et al.* [1999]).

3.1. A Bayesian Framework for Simulation Output Modeling

Under a Bayesian model, we assume that the simulation output $L(\theta_i, \omega_{ij})$ has a normal distribution with unknown mean J_i. The variance σ_i^2 can be known or unknown. After the simulation is performed, a posterior distribution of J_i, $p(J_i|L(\theta_i, \omega_{ij}), j = 1, 2, \ldots, N_i)$, can be constructed based on prior knowledge on two pieces of information: (i) prior knowledge of the system's performance, and (ii) current simulation output.

First, consider the case where the variance σ_i^2 is known. Further assume that the unknown mean J_i has the conjugate normal prior distribution $N(\eta_i, \nu_i^2)$. Under this assumption, the posterior distribution for any simulation output still belongs to the normal family (DeGroot [1970]). In particular, the posterior distribution of J_i is

$$p(J_i|L(\theta_i, \omega_{ij}), j = 1, 2, \ldots, N_i) \sim N\left(\frac{\sigma_i^2\eta_i + N_i v_i^2 \bar{J}_i}{\sigma_i^2 + N_i v_i^2}, \frac{\sigma_i^2 v_i^2}{\sigma_i^2 + N_i v_i^2}\right).$$

Suppose that the performance of any design is unknown before conducting the simulation, a non-informative prior can be applied and so v_i is an extremely large number. Then the posterior distribution of J_i is given by

$$p(J_i|L(\theta_i, \omega_{ij}), j = 1, 2, \ldots, N_i) \approx N\left(\bar{J}_i, \frac{\sigma_i^2}{N_i}\right).$$

If the variance σ_i^2 is unknown, a gamma-normal conjugate prior can be applied. Inoue and Chick [1998] show that the posterior distribution of J_i has a t distribution with mean $\bar{J}_i$, precision N_i/σ_i^2, and $N_i - 1$ degrees of freedom. In addition, DeGroot [1970] offers several other conjugate families of distributions for simulation samples from non-normal distributions, such as Poisson, negative binomial, and exponential distributions.

For ease of presentation, in this book we assume that the variance is known, and the simulation output samples $L(\theta_i, \omega_{ij})$ are normally distributed and independent from replication to replication (with mean J_i and variance σ_i^2), as well as independent across designs. The normality assumption is typically satisfied in simulation, because the output is obtained from an average performance or batch means (cf. Bechhofer *et al.* [1995]), so that Central Limit Theorem effects usually hold.

Assume that the unknown mean J_i has a conjugate normal prior distribution and consider non-informative prior distributions, which implies that no prior knowledge is available about the performance of any design before conducting the simulations. For ease of notation, let $\tilde{J}_i, i = 1, \ldots, k$, denote the random variable whose probability distribution is the posterior distribution of design i. Thus the posterior distribution of J_i is

$$\tilde{J}_i \sim N\left(\bar{J}_i, \frac{\sigma_i^2}{N_i}\right), \tag{3.1}$$

which implies that we are becoming more confident with the sample mean estimator when N_i increases as the variance decreases. This is consistent with the classical statistical model, in which as N_i increases, $\bar{J}_i$ becomes a better approximation to the true mean in the sense that its corresponding confidence interval becomes narrower.

After the simulation is performed, $\bar{J}_i$ can be calculated (σ_i^2 is approximated by the sample variance in practice). An advantage of Bayesian model is that we can say more about the unknown mean in a more straightforward way. For example, what is the probability that $\tilde{J}_i$ is less than some number a, i.e., $P\{\tilde{J}_i < a\}$? Another useful question is "what is the probability that design 1's mean is less than design 2's mean, i.e., $P\{\tilde{J}_1 < \tilde{J}_2\}$". With the normality assumption, the answers to these questions can be calculated easily.

Define

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}, \text{ standard normal probability density function, and}$$

$$\Phi(x) = \int_{-\infty}^{x} \varphi(t)dt, \text{ standard normal cumulative distribution function.}$$

Suppose the random variable X is normally distributed with mean μ and variance σ^2. For a constant a,

$$P\{X < a\} = P\left\{\frac{X-\mu}{\sigma} < \frac{a-\mu}{\sigma}\right\} = \Phi\left(\frac{a-\mu}{\sigma}\right), \quad \text{and} \tag{3.2}$$

$$P\{X > a\} = 1 - P\{X < a\} = 1 - \Phi\left(\frac{a-\mu}{\sigma}\right) = \Phi\left(\frac{\mu-a}{\sigma}\right). \tag{3.3}$$

In addition,

$$P\{X < 0\} = \Phi\left(\frac{-\mu}{\sigma}\right), \quad \text{and}$$

$$P\{X > 0\} = \Phi\left(\frac{\mu}{\sigma}\right).$$

When two designs are compared, say designs 1 and 2, since

$$\tilde{J}_2 - \tilde{J}_1 \sim N\left(\bar{J}_2 - \bar{J}_1, \frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}\right),$$

$$P\{\tilde{J}_1 < \tilde{J}_2\} = P\{\tilde{J}_2 - \tilde{J}_1 > 0\} = \Phi\left(\frac{\bar{J}_2 - \bar{J}_1}{\sqrt{\frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}}}\right). \tag{3.4}$$

Example 3.1. There are two designs. After conducting 4 independent simulation replications for each design, we obtain the following simulation output: $\bar{J}_1 = 10$, $\bar{J}_2 = 20$, $\sigma_1^2 = 100$, and $\sigma_2^2 = 81$. What is $P\{\tilde{J}_1 < 15\}$, $P\{\tilde{J}_2 > 15\}$, $P\{\tilde{J}_1 < 15 \text{ and } \tilde{J}_2 > 15\}$ and $P\{\tilde{J}_1 < \tilde{J}_2\}$?

Solution. Based on the simulation output, the posterior distributions for the unknown means are

$$\tilde{J}_1 \sim N(10, 25) \quad \text{and} \quad \tilde{J}_2 \sim N(20, 20.25).$$

Thus,

$$P\{\tilde{J}_1 < 15\} = \Phi\left(\frac{15-10}{\sqrt{25}}\right) = 0.841, \quad \text{and}$$

$$P\{\tilde{J}_2 > 15\} = \Phi\left(\frac{20-15}{\sqrt{20.25}}\right) = 0.867.$$

Since $\tilde{J}_1$ and $\tilde{J}_2$ are independent,

$$P\{\tilde{J}_1 < 15 \quad \text{and} \quad \tilde{J}_2 > 15\} = P\{\tilde{J}_1 < 15\}^* P\{\tilde{J}_2 > 15\} = 0.729.$$

On the other hand,

$$P\{\tilde{J}_1 < \tilde{J}_2\} = \Phi\left(\frac{10}{\sqrt{25+20.25}}\right) = 0.931.$$

□

Example 3.2. Continue Example 3.1. Suppose the estimated variances are smaller: $\sigma_1^2 = 40$, and $\sigma_2^2 = 36$. All other settings remain the same. What are $P\{\tilde{J}_1 < 15\}$, $P\{\tilde{J}_2 > 15\}$, $P\{\tilde{J}_1 < 15 \text{ and } \tilde{J}_2 > 15\}$, and $P\{\tilde{J}_1 < \tilde{J}_2\}$?

Solution. Since $\tilde{J}_1 \sim N(10, 10)$ and $\tilde{J}_2 \sim N(20, 9)$, we can calculate

$$P\{\tilde{J}_1 < 15\} = \Phi\left(\frac{15-10}{\sqrt{10}}\right) = 0.943,$$

$$P\{\tilde{J}_2 > 15\} = \Phi\left(\frac{20-15}{\sqrt{9}}\right) = 0.952,$$

$$P\{\tilde{J}_1 < 15 \text{ and } \tilde{J}_2 > 15\} = P\{\tilde{J}_1 < 15\} * P\{\tilde{J}_2 > 15\} = 0.898,$$

and

$$P\{\tilde{J}_1 < \tilde{J}_2\} = \Phi\left(\frac{10}{\sqrt{10+9}}\right) = 0.989.$$

□

Example 3.2 shows that smaller variance results in higher confidence on what we can infer about the unknown means which we try to estimate using simulation. Not surprisingly, the comparison probability between designs 1 and 2 also becomes higher.

To improve the simulation precision or enhance these probabilities, we have to increase the number of simulation replications. Another advantage of Bayesian framework is that we can easily estimate the sensitivity information about how these probabilities would change if we add some simulation replications *before* actually conducting the additional simulation replications.

Let Δ_i be a non-negative integer denoting the number of additional simulation replications we want to conduct for design i. If Δ_i is not too large, we can assume the sample statistics will not change too much after the additional simulations. Before the Δ_i simulation replications are performed, an approximation of the predictive posterior distribution for design i with Δ_i additional is

$$N\left(\bar{J}_i, \frac{\sigma_i^2}{N_i + \Delta_i}\right). \tag{3.5}$$

The approximation of the predictive posterior distribution can be used to assess the sensitivity information before additional simulations are actually performed. Example 3.3 demonstrates this approach.

Example 3.3. Continue Example 3.1. Suppose we want to conduct additional 16 simulation replications. Before the simulations are conducted, we want to estimate how those probabilities would be changed. Consider the following three scenarios: (i) add all the 16 replications to design 1, (ii) add all the 16 replications to design 2, and (iii) equally split the 16 replications between designs 1 and 2, i.e., 8 replications for each. For each scenario, please use the approximation of the predictive posterior distribution in Equation (3.5) to estimate $P\{\tilde{J}_1 < 15\}$, $P\{\tilde{J}_2 > 15\}$, $P\{\tilde{J}_1 < 15 \text{ and } \tilde{J}_2 > 15\}$, and $P\{\tilde{J}_1 < \tilde{J}_2\}$, if the additional simulations are performed?

Solution. In scenario (i), only design 1 receives additional 16 replications. Therefore the approximate predictive posterior distribution for design 1 becomes

$$\tilde{J}_1 \sim N\left(10, \frac{100}{4+16}\right).$$

Thus,

$$P\{\tilde{J}_1 < 15\} = \Phi\left(\frac{15-10}{\sqrt{5}}\right) = 0.987,$$

which is higher than that before adding the 16 additional replications. Since no additional simulation budget is allocated to design 2, $P\{\tilde{J}_2 > 15\}$ remains the same as that in Example 3.1. So

$$P\{\tilde{J}_1 < 15 \text{ and } \tilde{J}_2 > 15\} = 0.719,$$

and

$$P\{\tilde{J}_1 < \tilde{J}_2\} = \Phi\left(\frac{20-10}{\sqrt{\frac{100}{4+16} + \frac{81}{4}}}\right) = 0.977,$$

which is also higher than that in Example 3.1.

Similarly, we can calculate the probabilities for scenario (ii) and (iii). Specifically,

$$P\{\tilde{J}_1 < \tilde{J}_2\} = \Phi\left(\frac{20-10}{\sqrt{\frac{100}{4} + \frac{81}{4+16}}}\right) = 0.968$$

in scenario (i), and

$$P\{\tilde{J}_1 < \tilde{J}_2\} = \Phi\left(\frac{20-10}{\sqrt{\frac{100}{4+8} + \frac{81}{4+8}}}\right) = 0.995$$

in scenario (iii), which is the highest one among these three scenarios. Details of the results are presented in Table 3.1.

□

Example 3.3 shows that the resulting probabilities are different if we allocate the computing budget differently. As discussed later in

Table 3.1. Predictive probabilities in Example 3.3.

Δ_1	Δ_2	$P\{\tilde{J}_1 < 15\}$	$P\{\tilde{J}_2 > 15\}$	$P\{\tilde{J}_1 < 15 \text{ and } \tilde{J}_2 > 15\}$	$P\{\tilde{J}_1 < \tilde{J}_2\}$
0	0	0.841	0.867	0.729	0.931
16	0	0.987	0.867	0.855	0.977
0	16	0.841	0.994	0.836	0.968
8	8	0.958	0.973	0.932	0.995

the chapter, we want to allocate the simulation replications in a way that the comparison probability is maximized. In that sense, scenario (iii) is better than the other two scenarios in the Example 3.3 because it results in a higher $P\{\tilde{J}_1 < \tilde{J}_2\}$.

3.2. Probability of Correct Selection

After performing N_i simulation replications for design i, J_i is estimated using its sample mean $\bar{J}_i$ based on simulation output. Since we want to choose a design with minimum mean, it is a good idea to choose the design with minimum sample mean, i.e.,

$$b = \arg\min_i \bar{J}_i.$$

Design b is usually called an observed best design. Since the sample mean estimator has some variability, design b is not necessarily the one with the smallest unknown mean performance even though it has the smallest sample mean.

Define the *probability of correct selection* ($P\{\text{CS}\}$) as the probability that design b is actually the best design (i.e., with the smallest mean, hence the true best design).

$$\begin{aligned} P\{\text{CS}\} &= P\{\text{design } b \text{ is actually the best design}\} \\ &= P\{J_b < J_i, i \neq b | L(\theta_i, \xi_{ij}), j = 1, \ldots, N_i, i = 1, 2, \ldots, k\}. \end{aligned} \tag{3.6}$$

Using Equation (3.1) to simplify the notation, we rewrite Equation (3.6) as

$$P\{\text{CS}\} = P\{\tilde{J}_b < \tilde{J}_i, i \neq b\}. \tag{3.7}$$

Since $\tilde{J}_i \sim N(\bar{J}_i, \frac{\sigma_i^2}{N_i})$, as N_i increases, the variance of $\tilde{J}_i$ decreases, and so $P\{\text{CS}\}$ increases. The next question is how we can estimate $P\{\text{CS}\}$. As shown in Section 3.1, $P\{\text{CS}\}$ can be estimated easily if there are only 2 designs. When there are more than 2 designs, $P\{\text{CS}\}$ can then be estimated using a Monte Carlo simulation. However, estimating $P\{\text{CS}\}$ via Monte Carlo simulation is time-consuming.

Since the purpose of budget allocation is to improve simulation efficiency, we need a relatively fast and inexpensive way of estimating

$P\{\text{CS}\}$ within the budget allocation procedure. Efficiency is more crucial than estimation accuracy in this setting. We present two approximations which will serve as the basis for both methodological and theoretical development of computing budget allocation schemes. We refer to either one as the *Approximate Probability of Correct Selection* (*APCS*). These two different forms of *APCS* are developed using common approximation procedures used in simulation and statistics literature [Brately *et al.*, 1987; Chick, 1997; Law and Kelton, 2000]. They are

- *APCS-B* refers to the approximation using the Bonferroni inequality; and
- *APCS-P* refers to the approximation using a product form.

Specifics and their properties are given in the following lemmas.

Lemma 3.1. *The approximate probability of correct selection using the Bonferroni inequality*

$$APCS\text{-}B \equiv 1 - \sum_{i=1,i\neq b}^{k} P\{\tilde{J}_b > \tilde{J}_i\}$$

is a lower bound of $P\{\text{CS}\}$.

Proof. Let Y_i be a random variable. According to the Bonferroni inequality, $P\{\bigcap_{i=1}^{k}(Y_i < 0)\} \geq 1 - \sum_{i=1}^{k}[1 - P\{Y_i < 0\}]$. In our case, Y_i is replaced by $(\tilde{J}_b - \tilde{J}_i)$ to provide a lower bound for the probability of correct selection. That is,

$$\begin{aligned} P\{\text{CS}\} &= P\left\{\bigcap_{i=1,i\neq b}^{k} (\tilde{J}_b - \tilde{J}_i < 0)\right\} \\ &\geq 1 - \sum_{i=1,i\neq b}^{k} [1 - P\{\tilde{J}_b - \tilde{J}_i < 0\}] \\ &= 1 - \sum_{i=1,i\neq b}^{k} P\{\tilde{J}_b > \tilde{J}_i\} = APCS\text{-}B. \end{aligned}$$

□

APCS-B is much simpler to estimate. Since *APCS-B* is a lower bound of $P\{\text{CS}\}$, we are sure that $P\{\text{CS}\}$ is sufficiently high when *APCS-B* is satisfactory. Similarly, we will show that *APCS-P* is also a lower bound of $P\{\text{CS}\}$ in Lemma 3.2.

Lemma 3.2. *The approximate probability of correct selection in a product form*

$$APCS\text{-}P \equiv \prod_{i=1, i\neq b}^{k} P\{\tilde{J}_b < \tilde{J}_i\}$$

is a lower bound of $P\{\text{CS}\}$*.*

Proof. For a complete and rigorous proof, please see Chen [1996]. Here we give a quick intuitive explanation using a simple case where $k = 3$ and $b = 1$, i.e., there are 3 designs and design 1 is selected because it has the smallest sample mean.

$$\begin{aligned} P\{\text{CS}\} &= P\{\tilde{J}_1 < \tilde{J}_2 \text{ and } \tilde{J}_1 < \tilde{J}_3\} = P\{\tilde{J}_1 < \min(\tilde{J}_2, \tilde{J}_3)\} \\ &\geq P\{\tilde{J}_1 < \min(\tilde{J}_2, \tilde{J}_3)\}^* P\{\tilde{J}_1 < \max(\tilde{J}_2, \tilde{J}_3)\} \\ &\approx P\{\tilde{J}_1 < \tilde{J}_2\}^* P\{\tilde{J}_1 < \tilde{J}_3\} = APCS\text{-}P. \end{aligned}$$

□

Lemma 3.3. *Let* $N_i = \alpha_i T$*. If* $\alpha_i > 0$ *for all* i*, as* $T \to \infty$*, both APCS-B and APCS-P are asymptotically close to* $P\{\text{CS}\}$*.*

Proof. This lemma can be easily established. Note that

$$\tilde{J}_b - \tilde{J}_i \sim N\left(\delta_{bi}, \frac{\sigma_b^2}{N_b} + \frac{\sigma_i^2}{N_i}\right).$$

As $T \to \infty$, $N_i \to \infty$ for i. Since $\delta_{bi} < 0$,

$$P\{\tilde{J}_b > \tilde{J}_i\} = P\{\tilde{J}_b - \tilde{J}_i > 0\} \to 0,$$

and

$$P\{\tilde{J}_b < \tilde{J}_i\} = 1 - P\{\tilde{J}_b > \tilde{J}_i\} \to 1.$$

Therefore *APCS-B* $\to 1.0$ and *APCS-B* $\to 1.0$. Since both *APCS*s are lower bounds of $P\{\text{CS}\}$ and $P\{\text{CS}\} \leq 1.0$, either *APCS* is asymptotically close to $P\{\text{CS}\}$. □

Lemma 3.4. *Define $P\{\text{IS}\}$ as the probability of incorrect selection which equals $1-P\{\text{CS}\}$. Let $N_i = \alpha_i T$. If $\alpha_i > 0$ for all i, as $T \to \infty$, $(k-1)\max_i P\{\tilde{J}_b > \tilde{J}_i\} \geq P\{\text{IS}\} \geq \max_i P\{\tilde{J}_b > \tilde{J}_i\}$.*

Proof. See Glynn and Juneja [2004]. □

Lemma 3.4 implies that the probability of missing the best design converges to 0 as $T \to \infty$, provided that each design receives a non-zero proportion of computing budget. Also, the convergence rate of $P\{\text{IS}\}$ is the same as that of the slowest term of $P\{\tilde{J}_b > \tilde{J}_i\}$.

APCS can be computed very easily and quickly; no extra Monte Carlo simulation is needed. Numerical tests show that the *APCS* approximation can still lead to highly efficient procedures (e.g., Chen [1996]; Inoue and Chick [1998]; Inoue *et al.* [1999]). Furthermore, sensitivity information on the $P\{\text{CS}\}$ with respect to the number of simulation replications, N_i, which is central to the allocation of the computing budget — can be easily obtained. *APCS* is therefore used to approximate $P\{\text{CS}\}$ within the budget allocation procedure.

Example 3.4. Suppose there are three designs and 8 simulations replications are conducted for each design. Based on the simulation output, the estimated sample means are 10, 20, and 22, and their variances are 100, 81, and 144. What are *APCS-B* and *APCS-P*?

Solution. Based on the simulation output, $b = 1$ and the posterior distributions for the unknown means are

$$\tilde{J}_1 \sim N\left(10, \frac{100}{8}\right),$$

$$\tilde{J}_2 \sim N\left(20, \frac{81}{8}\right), \quad \text{and}$$

$$\tilde{J}_3 \sim N\left(22, \frac{144}{8}\right),$$

Applying Equation (3.4), we have

$$P\{\tilde{J}_1 < \tilde{J}_2\} = \Phi\left(\frac{20-10}{\sqrt{\frac{100}{8}+\frac{81}{8}}}\right) = 0.9822,$$

and

$$P\{\tilde{J}_1 < \tilde{J}_3\} = \Phi\left(\frac{22 - 10}{\sqrt{\frac{100}{8} + \frac{144}{8}}}\right) = 0.9851.$$

Therefore,

$$APCS\text{-}P = P\{\tilde{J}_1 < \tilde{J}_2\}^* P\{\tilde{J}_1 < \tilde{J}_3\} = 0.9822^* 0.9851 = 0.9676,$$

and

$$\begin{aligned} APCS\text{-}B &= 1 - (1 - P\{\tilde{J}_1 < \tilde{J}_2\}) - (1 - P\{\tilde{J}_1 < \tilde{J}_3\}) \\ &= 1 - 0.0178 - 0.0149 = 0.9673. \end{aligned}$$

□

3.3. Maximizing the Probability of Correct Selection

Based on the simulation output, the sample means for all designs that can be estimated, design b can be selected, and then an approximation of the probability that design b is actually the best design can also be estimated using an *APCS*. As N_i increases, $\bar{J}_i$ becomes a better approximation to J_i ($\equiv J(\theta_i)$) in the sense that the variance of the posterior distribution decreases, and so $P\{\text{CS}\}$ increases. As motivated in Chapter 2, instead of equally simulating all designs (i.e., equally increasing N_i), we will determine the best numbers of simulation replications for each design. Stating this more precisely, we wish to choose $N_1, N_2, \ldots, N_k$ such that $P\{\text{CS}\}$ is maximized, subject to a limited computing budget T,

(**OCBA–PCS Problem**)

$$\max_{N_1,\ldots,N_k} P\{\text{CS}\}$$

$$\text{s.t. } N_1 + N_2 + \cdots + N_k = T \quad \text{and} \quad N_i \geq 0. \tag{3.8}$$

Here $N_1 + N_2 + \cdots + N_k$ denotes the total computational cost. This formulation implicitly assumes that the computational cost of each replication is constant across designs. More general cases will be presented in Section 3.5.

We first consider the simple case that $k = 2$, where the theoretical optimal solution to problem (3.8) can be derived. Without loss of generality, assume $\bar{J}_1 < \bar{J}_2$, i.e., $b = 1$. The OCBA problem in (3.8) becomes

$$\max_{N_1,N_2} P\{\tilde{J}_1 < \tilde{J}_2\}$$
$$\text{s.t. } N_1 + N_2 = T. \tag{3.9}$$

As discussed in Section 3.1,

$$\tilde{J}_2 - \tilde{J}_1 \sim N\left(\bar{J}_2 - \bar{J}_1, \frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}\right),$$

yields

$$P\{\tilde{J}_1 < \tilde{J}_2\} = \Phi\left(\frac{\bar{J}_2 - \bar{J}_1}{\sqrt{\frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}}}\right).$$

Since Φ is a monotonically increasing function and $\bar{J}_2 - \bar{J}_1 > 0$, to maximize $\Phi\left(\frac{\bar{J}_2 - \bar{J}_1}{\sqrt{\frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}}}\right)$, we have to minimize $\frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}$. The optimization problem in (3.9) is equivalent to

$$\min_{N_1,N_2} \frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{N_2}$$
$$\text{s.t. } N_1 + N_2 = T.$$

Plugging the constraint that $N_2 = T - N_1$, we have

$$\min_{N_1} \frac{\sigma_1^2}{N_1} + \frac{\sigma_2^2}{T - N_1}. \tag{3.10}$$

The optimal solution to problem (3.10) can be obtained as follows

$$N_1 = \frac{\sigma_1 T}{\sigma_1 + \sigma_2}.$$

Thus

$$N_2 = T - N_1 = \frac{\sigma_2 T}{\sigma_1 + \sigma_2}.$$

The results are summarized in Theorem 3.1.

Theorem 3.1. *Given a total number of simulation samples T to be allocated to 2 competing designs whose variances are σ_1^2 and σ_2^2, respectively, the $P\{\text{CS}\}$ can be maximized when*

$$\frac{N_1}{N_2} = \frac{\sigma_1}{\sigma_2}.$$

□

Note that the optimal computing budget allocation for the 2-design case depends only on the standard deviations, but not on the means. For more general cases (i.e., $k > 2$), the $P\{\text{CS}\}$ formulation quickly becomes complicated and intractable. While we can still estimate $P\{\text{CS}\}$ via Monte Carlo simulation, it is very time-consuming. Since the purpose of budget allocation is to improve simulation efficiency, we need a relatively fast and inexpensive way of estimating $P\{\text{CS}\}$ within the budget allocation procedure. Efficiency is more crucial than estimation accuracy in this setting. Using the approximation given by Lemma 3.1, we consider the following approximation OCBA problem:

$$\max_{N_1,\ldots,N_k} 1 - \sum_{i=1,i\neq b}^{k} P\{\tilde{J}_b > \tilde{J}_i\}$$

$$\text{s.t. } \sum_{i=1}^{k} N_i = T \quad \text{and} \quad N_i \geq 0. \tag{3.11}$$

In the next section, an asymptotic allocation rule with respect to the number of simulation replications, N_i, will be presented.

3.3.1. *Asymptotically optimal solution*

Define α_i as the proportion of the total computing budget allocated to design i. Thus, $N_i = \alpha_i T$, and $\sum_{i=1}^{k} \alpha_i = 1$. First, we assume the variables, N_is, are continuous. Second, our strategy is to tentatively ignore all non-negativity constraints; all N_is can therefore assume any real value. Before the end of this section, we will show how all α_is become positive and hence all N_is are positive. Based on this idea, we first consider the following:

$$\max_{N_1,\ldots,N_k} 1 - \sum_{i=1,i\neq b}^{k} P\{\tilde{J}_b > \tilde{J}_i\}$$

$$\text{s.t. } \sum_{i=1}^{k} N_i = T. \tag{3.12}$$

Note that under normality assumption

$$\tilde{J}_i \sim N\left(\bar{J}_i, \frac{\sigma_i^2}{N_i}\right).$$

For the objective function,

$$\sum_{i=1,i\neq b}^{k} P\{\tilde{J}_b > \tilde{J}_i\} = \sum_{\substack{i=1\\ i\neq b}}^{k} \int_0^{\infty} \frac{1}{\sqrt{2\pi}\sigma_{b,i}} e^{-\frac{(x-\delta_{b,i})^2}{2\sigma_{b,i}^2}}\, dx$$

$$= \sum_{\substack{i=1\\ i\neq b}}^{k} \int_{-\frac{\delta_{b,i}}{\sigma_{b,i}}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\, dt,$$

where a new variable is introduced,

$$\sigma_{b,i}^2 = \frac{\sigma_b^2}{N_b} + \frac{\sigma_i^2}{N_i},$$

for notation simplification. Then, let F be the Lagrangian relaxation of Equation (3.12):

$$F = 1 - \sum_{\substack{i=1\\ i\neq b}}^{k} \int_{-\frac{\delta_{b,i}}{\sigma_{b,i}}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\, dt - \lambda \left(\sum_{i=1}^{k} N_i - T \right). \tag{3.13}$$

To find the stationary point of the Lagrangian function, we set $\frac{\partial F}{\partial N_i} = 0$ for each value of i:

$$\frac{\partial F}{\partial N_i} = \frac{\partial F}{\partial\left(-\frac{\delta_{b,i}}{\sigma_{b,i}}\right)} \frac{\partial\left(-\frac{\delta_{b,i}}{\sigma_{b,i}}\right)}{\partial \sigma_{b,i}} \frac{\partial \sigma_{b,i}}{\partial N_i} - \lambda$$

$$= \frac{-1}{2\sqrt{2\pi}} \exp\left[\frac{-\delta_{b,i}^2}{2\sigma_{b,i}^2}\right] \frac{\delta_{b,i}\sigma_i^2}{N_i^2(\sigma_{b,i}^2)^{3/2}} - \lambda = 0, \tag{3.14}$$

for $i = 1, 2, \ldots, k$, and $i \neq b$.

$$\frac{\partial F}{\partial N_b} = \frac{-1}{2\sqrt{2\pi}} \sum_{\substack{i=1\\ i\neq b}}^{k} \exp\left[\frac{-\delta_{b,i}^2}{2\sigma_{b,i}^2}\right] \frac{\delta_{b,i}\sigma_b^2}{N_b^2(\sigma_{b,i}^2)^{3/2}} - \lambda = 0, \tag{3.15}$$

We now examine the relationship between N_b and N_i for $i = 1, 2, \ldots, k$, and $i \neq b$. From Equation (3.14),

$$\frac{-1}{2\sqrt{2\pi}} \exp\left[\frac{-\delta_{b,i}^2}{2\sigma_{b,i}^2}\right] \frac{\delta_{b,i}}{(\sigma_{b,i}^2)^{3/2}} = -\lambda \frac{N_i^2}{\sigma_i^2}, \tag{3.16}$$

for $i = 1, 2, \ldots, k$, and $i \neq b$. Plugging Equation (3.16) into Equation (3.15), we have

$$\sum_{\substack{i=1\\ i\neq b}}^{k} \frac{-\lambda N_i^2 \sigma_b^2}{N_b^2 \sigma_i^2} - \lambda = 0.$$

Canceling out λ and rearranging terms yields that

$$N_b = \sigma_b \sqrt{\sum_{i=1, i\neq b}^{k} \frac{N_i^2}{\sigma_i^2}} \quad \text{or} \quad \alpha_b = \sigma_b \sqrt{\sum_{i=1, i\neq b}^{k} \frac{\alpha_i^2}{\sigma_i^2}}. \tag{3.17}$$

We further investigate the relationship between N_i and N_j, for any i, $j \in \{1, 2, \ldots, k\}$, and $i \neq j \neq b$. From Equation (3.14),

$$\exp\left(\frac{-\delta_{b,i}^2}{2\left(\frac{\sigma_b^2}{N_b} + \frac{\sigma_i^2}{N_i}\right)}\right) \cdot \frac{\delta_{b,i}\sigma_i^2/N_i^2}{\left(\frac{\sigma_b^2}{N_b} + \frac{\sigma_i^2}{N_i}\right)^{3/2}}$$
$$= \exp\left(\frac{-\delta_{b,j}^2}{2\left(\frac{\sigma_b^2}{N_b} + \frac{\sigma_j^2}{N_j}\right)}\right) \cdot \frac{\delta_{b,j}\sigma_j^2/N_j^2}{\left(\frac{\sigma_b^2}{N_b} + \frac{\sigma_j^2}{N_j}\right)^{3/2}}. \tag{3.18}$$

To reduce the total simulation time for identifying a good design, it is worthwhile to concentrate the computational effort on good designs. Namely, N_b should be increased relative to N_i, for $i = 1, 2, \ldots, k$, and $i \neq b$. This is indeed the case in the actual simulation experiments. And this assumption can be supported by considering a special case in Equation (3.17): When $\sigma_1 = \sigma_2 = \cdots = \sigma_k$,

$$N_b = \sqrt{\sum_{i=1, i\neq b}^{k} N_i^2}.$$

Therefore, we assume that $N_b \gg N_i$, which enables us to simplify Equation (3.18) as

$$\exp\left(\frac{-\delta_{b,i}^2}{2\left(\frac{\sigma_i^2}{N_i}\right)}\right) \cdot \frac{\delta_{b,i}\sigma_i^2/N_i^2}{\left(\frac{\sigma_i^2}{N_i}\right)^{3/2}} = \exp\left(\frac{-\delta_{b,j}^2}{2\left(\frac{\sigma_j^2}{N_j}\right)}\right) \cdot \frac{\delta_{b,j}\sigma_j^2/N_j^2}{\left(\frac{\sigma_j^2}{N_j}\right)^{3/2}}.$$

On rearranging terms, the above equation becomes

$$\exp\left(\frac{1}{2}\left(\frac{\delta_{b,j}^2}{\frac{\sigma_j^2}{N_j}} - \frac{\delta_{b,i}^2}{\frac{\sigma_i^2}{N_i}}\right)\right)\frac{N_j^{1/2}}{N_i^{1/2}} = \frac{\delta_{b,j}\sigma_i}{\delta_{b,i}\sigma_j}.$$

Taking the natural log on both sides, we have

$$\frac{\delta_{b,j}^2}{\sigma_j^2}N_j + \log(N_j) = \frac{\delta_{b,i}^2}{\sigma_i^2}N_i + \log(N_i) + 2\log\left(\frac{\delta_{b,j}\sigma_i}{\delta_{b,i}\sigma_j}\right),$$

or

$$\frac{\delta_{b,j}^2}{\sigma_j^2}\alpha_j T + \log(\alpha_j T) = \frac{\delta_{b,i}^2}{\sigma_i^2}\alpha_i T + \log(\alpha_i T) + 2\log\left(\frac{\delta_{b,j}\sigma_i}{\delta_{b,i}\sigma_j}\right),$$

which yields

$$\frac{\delta_{b,j}^2}{\sigma_j^2}\alpha_j T + \log(\alpha_j) = \frac{\delta_{b,i}^2}{\sigma_i^2}\alpha_i T + \log(\alpha_i) + 2\log\left(\frac{\delta_{b,j}\sigma_i}{\delta_{b,i}\sigma_j}\right). \quad (3.19)$$

To further facilitate the computations, we intend to find an asymptotic allocation rule. Namely, we consider the case that $T \to \infty$. While it is impossible to have an infinite computing budget in real life, our allocation rule provides a simple means for allocating simulation budget in a way that the efficiency can be significantly improved, as we will demonstrate in numerical testing later. As $T \to \infty$, all the log terms become much smaller than the other terms and are negligible. This implies

$$\frac{\delta_{b,j}^2}{\sigma_j^2}\alpha_j = \frac{\delta_{b,i}^2}{\sigma_i^2}\alpha_i.$$

Therefore, we obtain the ratio between α_i and α_j or between N_i and N_j as:

$$\frac{N_i}{N_j} = \frac{\alpha_i}{\alpha_j} = \left(\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}}\right)^2 \quad \text{for } i = 1, 2, \ldots, k, \text{ and } i \neq j \neq b. \quad (3.20)$$

Now we return to the issue of non-negative constraint for N_i, which we temporarily ignored. Note that from Equations (3.17) and (3.20), all α_is have the same sign. Since $\sum_{i=1}^{k} \alpha_i = 1$ and $N_i = \alpha_i T$, it implies that all $\alpha_i\text{s} \geq 0$, and hence $N_i\text{s} \geq 0$, where $i = 1, 2, \ldots, k$.

In conclusion, if a solution satisfies Equations (3.17) and (3.20), then it is a stationary point of the Lagrangian function in Equation (3.13) and so is a local optimal solution to the OCBA problem in Equation (3.11). We therefore have the following result:

Theorem 3.2. *Given a total number of simulation samples T to be allocated to k competing designs whose performance is depicted by random variables with means $J_1, J_2, \ldots, J_k$, and finite variances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_k^2$ respectively, as $T \to \infty$, the* Approximate Probability of Correct Selection (APCS) *can be asymptotically maximized when*

$$\frac{N_i}{N_j} = \left(\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}}\right)^2, \quad i, j \in \{1, 2, \ldots, k\}, \text{ and } i \neq j \neq b,$$

$$N_b = \sigma_b \sqrt{\sum_{i=1, i\neq b}^{k} \frac{N_i^2}{\sigma_i^2}},$$

where N_i is the number of simulation replications allocated to design i, $\delta_{b,i} = \bar{J}_b - \bar{J}_i$, and $\bar{J}_b \leq \min_{i\neq b} \bar{J}_i$. □

Remark 3.1. In the simple case that $k = 2$ and $b = 1$, Theorem 3.2 yields

$$N_1 = \sigma_1 \sqrt{\frac{N_2^2}{\sigma_2^2}}.$$

Therefore,

$$\frac{N_1}{N_2} = \frac{\sigma_1}{\sigma_2}.$$

□

The evaluated results in Remark 3.1 show that the allocation using Theorem 3.2 is identical to the theoretical optimal allocation solution for $k = 2$, where the closed-form optimal allocation can be found as shown in Theorem 3.1.

Remark 3.2. To gain better insight into this approach, consider another case where $k = 3$ and $b = 1$, Theorem 3.2 yields

$$\frac{N_2}{N_3} = \frac{\sigma_2^2 \delta_{1,3}^2}{\sigma_3^2 \delta_{1,2}^2},$$

which is consistent with our intuition explained using a hypothetical example depicted in Figure 3.1.

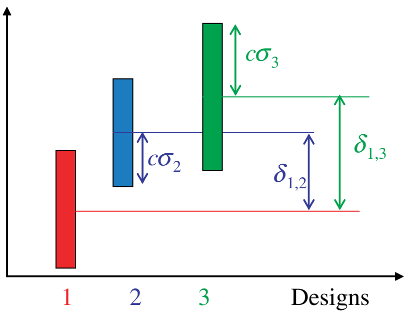


Figure 3.1. A hypothetical three-design example with some preliminary simulation, where design 1 has the smallest sample mean.

We check how the number of simulation samples for design 2, N_2, is affected by different factors. For a minimization problem, when $\bar{J}_3$ increases (we are more confident of the difference between design 1 and design 3) or σ_2 increases (we are less certain about design 2), N_2 increases as well. On the other hand, when $\bar{J}_2$ increases (we are more confident of the difference between design 1 and design 2) or σ_3 increases (we are less certain about design 3), N_2 decreases. The above relationship between N_2 and other factors is consistent with our intuition.

$\square$

Remark 3.3. Following Remark 3.2, we see that the allocated computing budget is proportional to variance and inversely proportional to the difference from the best design. In fact, we can rewrite the relationship between N_2 and N_3 as follows.

$$\frac{N_2}{N_3} = \left[\frac{\delta_{1,3}/\sigma_3}{\delta_{1,2}/\sigma_2}\right]^2 .$$

Intuitively, $\frac{\delta_{1,i}}{\sigma_i}$ can be considered as a signal to noise ratio for design i as compared with the observed best design. High $\frac{\delta_{1,i}}{\sigma_i}$ means that either the performance of design i is much worse than the performance of the best design or the estimation noise is small. In either cases, it means we are more confident in differentiating the design i from the best design, and so we do not need to allocate too much simulation budget to design i. Specifically, Theorem 3.2 shows that the allocated computing budget is inversely proportional to the square of the signal to noise ratio.

$\square$

Remark 3.4. The OCBA allocation given in Theorem 3.2 is derived by taking $T \to \infty$. While it is impossible to have an infinite computing budget in real life, our allocation rule provides a simple means for allocating simulation budget in a way that the efficiency can be significantly improved. Our numerical results presented in Chapter 4 indicate that the corresponding allocation in the OCBA procedure works very efficiently for small T, as well.

$\square$

Example 3.5. Suppose there are five alternative designs under consideration. We want to simulate all the 5 designs and then find the best design with minimum expected performance. With some preliminary simulation, we found that their sample means are 1, 2, 3, 4, and 5, and their sample variances are 1, 1, 9, 9, and 4. How should we allocate a total simulation budget of 50 replications if we apply OCBA given in Theorem 3.2? In addition, if we apply the notion of traditional two-stage procedures such as Rinott [1978] in which the number of replications is proportional to variance, how should we allocate these 50 replications to the five designs?

Solution. In this example, $b = 1$ and $\delta_{1,i} = 1, 2, 3,$ and 4, for $i = 2, 3, 4,$ and 5. $\sigma_i^2 = 1, 1, 9, 9,$ and 4 for $i = 1, 2, \ldots, 5$. Applying Theorem 3.2, we have

$$N_1 : N_2 : N_3 : N_4 : N_5 = 1.3 : 1 : 2.25 : 1 : 0.25.$$

Allocating the 50 simulation replications based on this ratio yields

$$N_1 = 11.2, N_2 = 8.6, N_3 = 19.4, N_4 = 8.6, \text{ and } N_5 = 2.2.$$

Since the simulation replication number should be an integer, those numbers are rounded to nearest integers with a constraint that the total must be 50, yielding the following OCBA allocation:

$$N_1 = 11, N_2 = 9, N_3 = 19, N_4 = 9, \text{ and } N_5 = 2.$$

On the other hand, if the simulation budget is allocated in a way that the simulation replication number is proportional to its variance, then

$$N_1 : N_2 : N_3 : N_4 : N_5 = 1 : 1 : 9 : 9 : 4.$$

Allocating the 50 simulation replications based on this ratio, we have

$$N_1 = 2.1, N_2 = 2.1, N_3 = 18.8, N_4 = 18.8, \text{ and } N_5 = 8.2.$$

Rounding them to nearest integers, we have

$$N_1 = 2, N_2 = 2, N_3 = 19, N_4 = 19, \text{ and } N_5 = 8.$$

□

The computing budget allocation based on variances like Rinott [1978] procedure was very popular. The OCBA allocation turns out to be very different. In Chapter 4, we will have more extensive empirical testing and comparison, where numerical results indicate that OCBA is much more efficient than traditional approaches.

3.3.2. *OCBA simulation procedure*

With Theorem 3.2, we now present a cost-effective sequential approach based on OCBA to select the best design from k alternatives with a given computing budget. Initially, n_0 simulation replications for each of k designs are conducted to get some information about the performance of each design during the first stage. As simulation proceeds, the sample means and sample variances of all designs are computed from the data already collected up to that stage. According to this collected simulation output, an incremental computing budget, Δ, is allocated based on Theorem 3.2 at each stage. Ideally, each new replication should bring us closer to the optimal solution. This procedure is continued until the total budget T is exhausted. The algorithm is summarized as follows.

OCBA Procedure (Maximizing $P\{\text{CS}\}$)

```
INPUT        $k, T, \Delta, n_0$ ($T$-$kn_0$ is a multiple of $\Delta$ and $n_0 \geq$
             5);
INITIALIZE   $l \leftarrow 0$;
             Perform $n_0$ simulation replications for all
             designs; $N_1^l = N_2^l = \cdots = N_k^l = n_0$.
LOOP         WHILE $\sum_{i=1}^{k} N_i^l < T$ DO
  UPDATE     Calculate sample means $\bar{J}_i = \frac{1}{N_i^l} \sum_{j=1}^{N_i^l}$
             $L(\theta_i, \omega_{ij})$, and sample standard deviations $s_i =$
```

$\sqrt{\frac{1}{N_i^l-1}\sum_{j=1}^{N_i^l}(L(\theta_i,\omega_{ij})-\bar{J}_i)^2}$, $i = 1,\ldots,k$, using the new simulation output; find $b = \arg\min_i \bar{J}_i$.

ALLOCATE Increase the computing budget by Δ and calculate the new budget allocation, N_1^{l+1}, $N_2^{l+1},\ldots,N_k^{l+1}$, according to

(1) $\frac{N_i^{l+1}}{N_j^{l+1}} = \left(\frac{s_i(\bar{J}_b-\bar{J}_j)}{s_j(\bar{J}_b-\bar{J}_i)}\right)^2$, for all $i \neq j \neq b$, and

(2) $N_b^{l+1} = s_b\sqrt{\sum_{i=1,i\neq b}^{k}\left(\frac{N_i^{l+1}}{s_i}\right)^2}$,

SIMULATE Perform additional $\max(N_i^{l+1} - N_i^l, 0)$ simulations for design i, $i = 1,\ldots,k$; $l \leftarrow l+1$.

END OF LOOP

In the above algorithm, l is the iteration number. σ_i^2 is be approximated by the sample variance s_i^2. As simulation evolves, design b, which is the design with the smallest sample mean, may change from iteration to iteration, although it will converge to the optimal design as the l goes to infinity. When b changes, Theorem 3.2 is still applied in the UPDATE step. However, the older design b may not be simulated at all in this iteration because design b is the one receiving more computing budget and the older one might have received more than what it should have in earlier iterations.

In the procedure, we need to select the initial number of simulations, n_0, and the one-time increment, Δ. It is well understood that n_0 cannot be too small as the estimates of the mean and the variance may be very poor, resulting in premature termination of the comparison. A suitable choice for n_0 is between 5 and 20 (Law and Kelton [1991]; Bechhofer *et al.* [1995]). Also, it is wise to avoid large Δ to prevent a poor allocation before a correction can be made in the next iteration, which is particularly important in the early stages. On the other hand, if Δ is small, we need to perform the ALLOCATE step many times. A suggested choice for Δ is a number bigger than 5 but smaller than 10% of the simulated designs. However, when the simulation is relatively expensive, the computation cost of the

ALLOCATION step becomes negligible. In this case, Δ should be as small as possible and can be set as 1. More extensive analysis and discussion will be given in Chapter 4.

3.4. Minimizing the Total Simulation Cost

A higher simulation budget usually results in a higher $P\{\text{CS}\}$. Instead of trying to maximize $P\{\text{CS}\}$ subject to a fixed computing budget, a decision-maker may want to minimize overall computing cost (total number of simulation replications) in order to achieve a desired $P\{\text{CS}\}$, such as 95%, i.e.,

(OCBA–SC Problem)

$$\begin{gathered}\min_{N_1,\ldots,N_k}[N_1+N_2+\cdots+N_k]\\ \text{s.t. } P\{\text{CS}\}\geq P^*,\\ N_i\in\Gamma,\quad i=1,\ldots,k.\end{gathered}\tag{3.21}$$

where P^* is the desired "confidence level." While the problem appears to be different from the OCBA–PCS problem given in Section 3.3, we will show that their approximate solutions are identical, so the same OCBA procedure applies to both problems [Chen and Yücesan, 2005]. This OCBA–SC problem can be viewed as the dual perspective, which essentially corresponds to the traditional statistical ranking and selection approach.

Using the *APCS-B* approximation given by Lemma 3.1, the computing budget allocation problem (3.21) becomes

$$\begin{gathered}\min_{N_1,\ldots,N_k}[N_1+N_2+\cdots+N_k]\\ \text{s.t. } 1-\sum_{i=1,i\neq b}^{k}P\{\tilde{J}_b>\tilde{J}_i\}\geq P^*,\quad\text{and}\\ N_i\in\Gamma,\ i=1,\ldots,k.\end{gathered}$$

By taking a similar approach as in Section 3.3, we assume the variables, N_is, are continuous. In addition, our strategy is to tentatively ignore all non-negativity constraints; all N_is can therefore assume any real value. Again, we will show how all N_is are positive before the end of this section. Based on this non-integral assumption, we first consider the following:

$$\min_{N_1,\ldots,N_k} [N_1 + N_2 + \cdots + N_k]$$

$$\text{s.t. } 1 - \sum_{\substack{i=1 \\ i\neq b}}^{k} \int_{-\frac{\delta_{b,i}}{\sigma_{b,i}}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt \geq P^*. \tag{3.22}$$

Let F be the Lagrangian of (3.22):

$$F = \sum_{i=1}^{k} N_i - \lambda \left(1 - \sum_{\substack{i=1 \\ i\neq b}}^{k} \int_{-\frac{\delta_{b,i}}{\sigma_{b,i}}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt - P^* \right).$$

To find the stationary point of the Lagrangian function, we set $\frac{\partial F}{\partial N_i} = 0$ for each value of i:

$$\frac{\partial F}{\partial N_i} = 1 + \frac{\lambda}{2\sqrt{2\pi}} \exp\left[\frac{-\delta_{b,i}^2}{2\sigma_{b,i}^2} \right] \frac{\delta_{b,i}\sigma_i^2}{N_i^2 (\sigma_{b,i}^2)^{3/2}} = 0,$$

$$\text{for } i = 1, 2, \ldots, k, \text{ and } i \neq b. \tag{3.23}$$

$$\frac{\partial F}{\partial N_b} = 1 + \sum_{\substack{i=1 \\ i\neq b}}^{k} \frac{\lambda}{2\sqrt{2\pi}} \exp\left[\frac{-\delta_{b,i}^2}{2\sigma_{b,i}^2} \right] \frac{\delta_{b,i}\sigma_b^2}{N_b^2 (\sigma_{b,i}^2)^{3/2}} = 0, \tag{3.24}$$

$$\lambda \left(1 - \sum_{\substack{i=1 \\ i\neq b}}^{k} \int_{-\frac{\delta_{b,i}}{\sigma_{b,i}}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt - P^* \right) = 0, \text{ and } \lambda \geq 0.$$

We now examine the relationship between N_b and N_i for $i = 1, 2, \ldots, k$, and $i \neq b$. From Equation (3.23),

$$\frac{\lambda}{2\sqrt{2\pi}} \exp\left[\frac{-\delta_{b,i}^2}{2\sigma_{b,i}^2} \right] \frac{\delta_{b,i}}{(\sigma_{b,i}^2)^{3/2}} = -\frac{N_i^2}{\sigma_i^2},$$

$$\text{for } i = 1, 2, \ldots, k, \text{ and } i \neq b. \tag{3.25}$$

Plugging Equation (3.25) into (3.24), we have

$$1 - \sum_{\substack{i=1 \\ i\neq b}}^{k} \frac{N_i^2 \sigma_b^2}{N_b^2 \sigma_i^2} = 0.$$

Then

$$N_b = \sigma_b \sqrt{\sum_{i=1, i \neq b}^{k} \frac{N_i^2}{\sigma_i^2}} \quad \text{or} \quad \alpha_b = \sigma_b \sqrt{\sum_{i=1, i \neq b}^{k} \frac{\alpha_i^2}{\sigma_i^2}}. \tag{3.26}$$

We further investigate the relationship between N_i and N_j, for any i, $j \in \{1, 2, \ldots, k\}$, and $i \neq j \neq b$. From Equation (3.23),

$$\begin{aligned}
&\exp\left(\frac{-\delta_{b,i}^2}{2\left(\frac{\sigma_b^2}{N_b} + \frac{\sigma_i^2}{N_i}\right)}\right) \cdot \frac{\delta_{b,i}\sigma_i^2/N_i^2}{\left(\frac{\sigma_b^2}{N_b} + \frac{\sigma_i^2}{N_i}\right)^{3/2}} \\
&\qquad = \exp\left(\frac{-\delta_{b,j}^2}{2\left(\frac{\sigma_b^2}{N_b} + \frac{\sigma_j^2}{N_j}\right)}\right) \cdot \frac{\delta_{b,j}\sigma_j^2/N_j^2}{\left(\frac{\sigma_b^2}{N_b} + \frac{\sigma_j^2}{N_j}\right)^{3/2}}.
\end{aligned} \tag{3.27}$$

Equation (3.27) is identical to Equation (3.18), from which we have derived an asymptotic optimal solution. Specifically, we consider an asymptotic case where $P^* \to 1$ and $APCS \to 1.0$, and so $T \to \infty$. Following the same derivation as in Section 3.3, as $T \to \infty$, we have

$$\frac{N_i}{N_j} = \frac{\alpha_i}{\alpha_j} = \left(\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}}\right)^2$$
$$\text{for} \quad i = 1, 2, \ldots, k, \ \text{and } i \neq j \neq b. \tag{3.28}$$

Now we return to the nonnegativity constraints for N_i, which we have temporarily ignored. Note that from Equation (3.26) and (3.28), all α_is have the same sign. Since $\sum_{i=1}^{k} \alpha_i = 1$ and $N_i = \alpha_i n$, it implies that all α_is $\geqslant 0$, and hence N_is ≥ 0, for $i = 1, 2, \ldots, k$.

In conclusion, if a solution satisfies Equation (3.26) and (3.28), then it is a local optimal solution to the OCBA–SC problem in Equation (3.21). We therefore have the following result:

Theorem 3.3. *Suppose there are k competing designs whose performance is depicted by random variables with means $J_1, J_2, \ldots, J_k$, and finite variances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_k^2$, respectively. Given a desired level of* Approximate Probability of Correct Selection (APCS), *P^*, the* *total number of simulation replications, $N_1 + N_2 + \cdots + N_k$, can be asymptotically minimized when*

(1) $\frac{N_i}{N_j} = \left(\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}}\right)^2$, $i, j \in \{1, 2, \ldots, k\}$, *and* $i \neq j \neq b$,

(2) $N_b = \sigma_b \sqrt{\sum_{i=1, i\neq b}^{k} \frac{N_i^2}{\sigma_i^2}}$.

□

The asymptotic allocation given in Theorem 3.3 for problem (3.21) is actually identical to the asymptotic solution given in Section 3.3, although these problems look different. In fact, they share the same objective of maximizing simulation efficiency. It is nice that they end up with the same solution. The simulation procedure is similar with the one described in Section 3.3 except the stopping criterion. Again, n_0 simulation replications for each of k designs are initially conducted to get some information about the performance of each design during the first stage. As simulation proceeds, the sample means and sample variances of each design are computed from the data already collected up to that stage. Then *APCS* can be calculated using Lemma 3.1. If *APCS* is not sufficiently high, an incremental computing budget, Δ, is allocated based on Theorem 3.3 at each stage. Ideally, each new replication should bring us closer to the optimal solution. This procedure is continued until the desired *APCS* level is achieved.

OCBA Procedure (Minimizing Simulation Cost)

```
INPUT         $k, P^*, \Delta, n_0 (n_0 \geq 5)$;
INITIALIZE    $l \leftarrow 0$;
              Perform $n_0$ simulation replications for all designs;
              $N_1^l = N_2^l = \cdots = N_k^l = n_0$.
LOOP
  UPDATE      Calculate sample means $\bar{J}_i = \frac{1}{N_i^l}\sum_{j=1}^{N_i^l} L(\theta_i, \omega_{ij})$,
              and sample standard deviation $s_i = \sqrt{\frac{1}{N_i^l - 1}\sum_{j=1}^{N_i^l}(L(\theta_i, \omega_{ij}) - \bar{J}_i)^2}$, $i = 1, \ldots, k$, using
              the new simulation output; compute, $i = 1, \ldots, k$;
```

```
            find $b = \arg\min_i \bar{J}_i$; calculate $APCS = 1 - \sum_{i=1,i\neq b}^{k} \Phi\left(\frac{\bar{J}_b - \bar{J}_i}{\sqrt{\frac{s_b^2}{N_b^l} + \frac{s_i^2}{N_i^l}}}\right)$.
    CHECK       If $APCS \geq P^*$, stop; Otherwise, continue.
    ALLOCATE    Increase the computing budget by $\Delta$ and calculate the new budget allocation, $N_1^{l+1}$, $N_2^{l+1}, \ldots, N_k^{l+1}$, according to
                (1) $\frac{N_i^{l+1}}{N_j^{l+1}} = \left(\frac{s_i(\bar{J}_b - \bar{J}_j)}{s_j(\bar{J}_b - \bar{J}_i)}\right)^2$, for all $i \neq j \neq b$, and
                (2) $N_b^{l+1} = s_b \sqrt{\sum_{i=1,i\neq b}^{k} \left(\frac{N_i^{l+1}}{s_i}\right)^2}$,
    SIMULATE    Perform additional $\max(N_i^{l+1} - N_i^l, 0)$ simulations for design $i$, $i = 1, \ldots, k$; $l \leftarrow l + 1$.
END OF LOOP
```

In the UPDATE step, the *APCS-B* in Lemma 3.1 is applied. An alternative nice approximation is *APCS-P* given in Lemma 3.2. Specifically,

$$APCS\text{-}P = \prod_{i=1,i\neq b}^{k} P\{\tilde{J}_b - \tilde{J}_i < 0\} = \prod_{i=1,i\neq b}^{k} \Phi\left(\frac{\bar{J}_i - \bar{J}_b}{\sqrt{\frac{s_i^2}{N_i^l} + \frac{s_b^2}{N_b^l}}}\right).$$

The numerical testing shows that both forms of *APCS* provide good approximation, particularly when N is not small. Since both are lower bounds of $P\{\text{CS}\}$, we are sure that $P\{\text{CS}\}$ is sufficiently high when any *APCS* is satisfactory.

3.5. Non-Equal Simulation Costs

In deriving the OCBA solution in previous sections, we assume that the computation costs of one simulation replication for different designs are roughly the same. In practice, some designs may have very different structures with different characteristics, resulting in very different computation times. In this section, we will extend the OCBA problem to cases where computation times are different among the candidate designs.

Let the computation cost per simulation run for design i be c_i. Of course, $c_i > 0$. We wish to choose $N_1, N_2, \ldots, N_k$ such that $P\{\text{CS}\}$ is maximized, subject to a limited computing budget T, i.e.,

$$\min_{N_1,\ldots,N_k} P\{\text{CS}\}$$

$$\text{s.t. } c_1N_1 + c_2N_2 + \cdots + c_kN_k = T \quad \text{and} \quad N_i \geq 0. \tag{3.29}$$

When $c_i = 1$ (or a constant), the computing budget allocation problem in Equation (3.29) degenerates into the special case considered in Section 3.3. Similarly, *APCS* is used to approximate $P\{\text{CS}\}$ using Lemma 3.1. Let $T_i = c_iN_i$. Thus T_i is the total computing budget allocated to design i and $N_i = T_i/c_i$ is the number of simulation replications for design i. The OCBA problem in Equation (3.29) can be rewritten as follows.

$$\max_{T_1,\ldots,T_k} 1 - \sum_{i=1,i\neq b}^{k} P\{\tilde{J}_b > \tilde{J}_i\}$$

$$\text{s.t. } \sum_{i=1}^{k} T_i = T \quad \text{and} \quad T_i \geq 0. \tag{3.30}$$

Since

$$\tilde{J}_i \sim N\left(\bar{J}_i, \frac{\sigma_i^2}{N_i}\right),$$

replacing the variable N_i with T_i yields

$$\tilde{J}_i \sim N\left(\bar{J}_i, \frac{(\sqrt{c_i}\sigma_i)^2}{T_i}\right). \tag{3.31}$$

Note that the budget allocation problem in Equation (3.30) is the same as that in Equation (3.12), except that the decision variable is T_i rather than N_i and the variance of $\tilde{J}_i$ is $\frac{(\sqrt{c_i}\sigma_i)^2}{T_i}$ rather than $\frac{\sigma_i^2}{N_i}$. Following the same derivations given in Section 3.3, an asymptotic optimal solution to problem (3.30) with respect to T_i can be obtained as follows.

$$T_b = \sqrt{c_b}\sigma_b \sqrt{\sum_{i=1,i\neq b}^{k} \frac{T_i^2}{c_i\sigma_i^2}},$$

and

$$\frac{T_i}{T_j} = \left(\frac{\sqrt{c_i}\sigma_i/\delta_{b,i}}{\sqrt{c_j}\sigma_j/\delta_{b,j}}\right)^2 \text{ for } i = 1, 2, \ldots, k, \text{ and } i \neq j \neq b. \tag{3.32}$$

Therefore, the asymptotic optimal solution with respect to N_i is summarized in the following theorem.

Theorem 3.4. *Suppose the computation cost for one simulation replication for design i is c_i. Given a total simulation budget T to be allocated to k competing designs, the* Approximate Probability of Correct Selection (APCS) *can be asymptotically maximized when*

$$\frac{N_i}{N_j} = \left(\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}}\right)^2, \quad i, j \in \{1, 2, \ldots, k\}, \text{ and } i \neq j \neq b,$$

$$N_b = \sigma_b \sqrt{\sum_{i=1, i\neq b}^{k} \frac{c_i N_i^2}{c_b \sigma_i^2}}.$$

□

It is interesting to note that the relationship between N_i and N_j in asymptotically optimal allocation remains the same no matter whether the computation costs are equal or not. By checking Equation (3.31), we see that increasing the simulation cost for one replication of design i by c times has a same impact as increasing the variance for design i by c times. In either case, the weight of computing budget allocation for design i is c times higher. However, if design i is c times more expensive to simulate, the ratio of N_i and N_j remains the same even with the additional computing budget allocated to design i.

3.6. Minimizing Opportunity Cost

In previous sections, we focus on the probability of correctly selecting the best design ($P\{\text{CS}\}$), which is the most commonly studied measure of performance. When the selection is incorrect, i.e., the best is not selected, it is not distinguished whether a mildly bad design or a very bad design is chosen. A second important measure of selection quality is the expected opportunity cost (E[OC]), which penalizes particularly bad choices more than mildly bad choices. For example, it may be better to be wrong 99% of the time if the penalty for being wrong is \$1 (an expected opportunity cost of $0.99 \times \$1 = \$0.99$) rather than being wrong only 1% of the time if the penalty is \$1,000 (an expected opportunity cost of $0.01 \times \$1{,}000 = \$10$).

In this section, we turn our attention to E[OC]. The opportunity cost is "the cost of any activity measured in terms of the best alternative forgone." (cf. [Chick and Inoue, 2001a; He *et al.* 2007]. In our setting, the opportunity cost is the difference between the unknown mean of the selected design b and the unknown mean of the actual best design (defined as i^* below). From the simulation efficiency perspective, one has the same question to ask: how should we control the simulation so that we can select a design within the given computing budget while the expected opportunity cost is minimized, instead of maximizing $P\{\text{CS}\}$ as in previous sections?

Deriving an asymptotic solution for minimizing E[OC] is much more complicated than its counterpart for $P\{\text{CS}\}$. We will present a greedy selection procedure to reduce the E[OC] of a potentially incorrect selection by taking a similar OCBA approach to selection. In addition, as shown in He *et al.* [2007], the OCBA procedure presented earlier performs very well not only for maximizing $P\{\text{CS}\}$, but also minimizing E[OC]. Therefore, to minimize E[OC], one can either use the greedy procedure presented in this section, or simply continue using the earlier OCBA procedures.

Our approach is to allocate simulation runs sequentially so that an approximation to E[OC] can be minimized when the simulations are finished. Following the same Bayesian model presented in Section 3.1, the expected opportunity cost (E[OC]) is defined as follows

$$\mathrm{E[OC]} = \mathrm{E}[\tilde{J}_b - \tilde{J}_{i*}] = \sum_{i=1, i\neq b}^{k} P\{i = i^*\}\mathrm{E}[\tilde{J}_b - \tilde{J}_{i*} | i = i^*], \quad (3.33)$$

where i^* is the true best design and the expectation is taken with respect to the posterior distribution of the $\tilde{J}_i$, for $i = 1, 2, \ldots, k$, given all simulations seen so far. According to the definition of the conditional expectation [Whittle, 2000],

$$E(X|A) = \frac{E[X \cdot I(A)]}{E[I(A)]} = \frac{E[X \cdot I(A)]}{P(A)},$$

where X is a random variable, and for an event A we define the *indicator function* to be

$$I(A) = \begin{cases} 1 & \text{if } A \text{ does occur.} \\ 0 & \text{if } A \text{ does not occur.} \end{cases}.$$

Therefore,

$$\mathrm{E[OC]} = \sum_{i=1, i \neq b}^{k} E[I(i = i^*) \times (\tilde{J}_b - \tilde{J}_i)]. \tag{3.34}$$

A convenient closed-form expression for E[OC] is unknown for all but a few special cases. We therefore present an approximation of E[OC] that is key for the new selection procedure.

Lemma 3.5. *A good approximation to $E[OC]$ is given by*

$$EEOC \equiv \sum_{i=1, i \neq b}^{k} P(\tilde{J}_i < \tilde{J}_b)^* E[\tilde{J}_b - \tilde{J}_i | \tilde{J}_i < \tilde{J}_b]. \tag{3.35}$$

We refer to this approximation as the Estimated Expected Opportunity Cost (EEOC). *In fact, EEOC is an upper bound of $E[OC]$.*

Proof. Note that

$$\begin{aligned} P(i = i^*) &= P(\tilde{J}_i < \tilde{J}_b)^* P(\tilde{J}_i < \tilde{J}_j \text{ for all } j \notin \{i, b\} | \tilde{J}_i < \tilde{J}_b) \\ &\leq P(\tilde{J}_i < \tilde{J}_b). \end{aligned}$$

Therefore,

$$\mathrm{E}[I(i = i^*) * (\tilde{J}_b - \tilde{J}_i)] \leq \mathrm{E}[I(\tilde{J}_i < \tilde{J}_b) * (\tilde{J}_b - \tilde{J}_i)].$$

Thus,

$$\begin{aligned} \mathrm{E[OC]} &= \sum_{i=1, i \neq b}^{k} E[I(i = i^*) \times (\tilde{J}_b - \tilde{J}_i)] \\ &\leq \sum_{i=1, i \neq b}^{k} E[I(\tilde{J}_i < \tilde{J}_b)^* (\tilde{J}_b - \tilde{J}_i)] \\ &= \sum_{i=1, i \neq b}^{k} P\{\tilde{J}_i < \tilde{J}_b\}^* \mathrm{E}[\tilde{J}_b - \tilde{J}_i | \tilde{J}_i < \tilde{J}_b] \equiv EEOC. \end{aligned}$$

Since design b has the best (smallest) sample mean so far, i.e., $\bar{J}_b < \bar{J}_i$ for all i, the conditional probability, $P\{\tilde{J}_i < \tilde{J}_j$ for all $j \notin \{i, b\}|$

$\tilde{J}_i < \tilde{J}_b\}$, will often be close to 1, so that $P\{\tilde{J}_{i*} < \tilde{J}_b\}$ is often a good approximation to $P(i = i^*)$. Similarly we hope $\mathrm{E}[I(\tilde{J}_i < \tilde{J}_b)(\tilde{J}_b - \tilde{J}_i)]$ would be a reasonable approximation to $\mathrm{E}[I(i = i^*)(\tilde{J}_b - \tilde{J}_i)]$. The numerical tests performed by He *et al.* [2007] show that using *EEOC* to approximate E[OC] leads to a highly efficient procedure. □

Since *EEOC* is an upper bound of E[OC], we are sure that E[OC] is sufficiently low when *EEOC* is satisfactory. We therefore use *EEOC* in the above equation to approximate E[OC] in our selection procedure. In terms of the approximation property, the term *EEOC* can be interpreted as the sum of expected opportunity costs for each of the $k - 1$ pairwise comparisons between the best and each alternative, given all of the sampling information that is available. This term therefore resembles the well-known Bonferroni bound for probabilities, which states that the probability that the best design is not correctly selected, is less than the sum of the probabilities that the best is not correctly selected in the $k - 1$ pairwise comparisons of the best with each alternative. The Bonferroni bound is tight when $k = 2$ or when the value of each term is small, and is known to become less tight as k increases. If only a few competing alternatives have a similar performance or if we have conducted extensive simulation on most competing designs, then the bound can be relatively tight.

Ideally we would choose $N_1, N_2, \ldots, N_k$ to minimize E[OC] or *EEOC*, given a limited computing budget. Even though *EEOC* is much easier to compute than E[OC], it is still intractable to derive a solution which optimizes *EEOC* as we did in earlier sections. Following the same notion of the greedy approach given in Chen *et al.* [1999] and Hsieh *et al.* [2001], we sequentially minimize *EEOC*.

Specifically, we allocate a few replications at each stage of a sequential procedure to reduce *EEOC* iteratively. A critical component in the proposed procedure is to estimate how *EEOC* changes as N_i changes. Let Δ_i be a non-negative integer denoting the number of additional simulation replications allocated to design i in the next stage of sampling. We are interested in assessing how *EEOC* would be affected if design i were simulated for Δ_i additional replications. In other words, we are interested in assessing how *promising* it is to simulate design i. This assessment must be made *before* actually conducting Δ_i simulation replications. According to the Bayesian model presented in Section 3.1, if we conduct Δ_i additional replications of design i, given a finite N_i, the posterior distribution for the unknown mean of design i is

$$N\left(\frac{1}{N_i+\Delta_i}\sum_{j=1}^{N_i+\Delta_i} L(\theta_i,\omega_{ij}), \frac{\sigma_i^2}{N_i+\Delta_i}\right),$$

where $L(\theta_i,\omega_{ij})$ is the j-th sample of design i. When Δ_i is relatively small compared to N_i, the difference between $\frac{1}{N_i}\sum_{j=1}^{N_i} L(\theta_i,\omega_{ij})$ and $\frac{1}{N_i+\Delta_i}\sum_{j=1}^{N_i+\Delta_i} L(\theta_i,\omega_{ij})$ will be small. A heuristic approach to the approximation of the predictive posterior distribution yields

$$\tilde{J}_i \sim N\left(\frac{1}{N_i}\sum_{j=1}^{N_i} L(\theta_i,\omega_{ij}), \frac{\sigma_i^2}{N_i+\Delta_i}\right) \text{ for design } i. \tag{3.36}$$

We therefore approximate the distribution of the unknown mean, given that Δ_i is small, by a $N(\frac{1}{N_i}\sum_{j=1}^{N_i} L(\theta_i,\omega_{ij}), \frac{\sigma_i^2}{N_i+\Delta_i})$ distribution. The *EEOC* can then be calculated by plugging Equation (3.36) into the *EEOC* formula in Equation (3.35). In particular, if we allocate Δ_i additional samples for design i (for any i, including design b), then the corresponding estimated expected opportunity cost, denoted as *EEOC*(i), is determined by using the original distributions for the unknown means of designs other than i, and using the modified distribution in Equation (3.36) for design i, to obtain

$$EEOC(i) = \sum_{j=1,j\neq b}^{k} \int_0^{+\infty} x f_{b,j,i}^*(x)\,dx, \quad \text{for } i = 1,\ldots,k; \tag{3.37}$$

where $f_{b,j,i}^*(x)$ is the PDF of the difference between design b and design j, given that Δ_i more replications are given to design i, and none is allocated to the others, for any i and $j \neq b$. If $i = b$, then $f_{b,j,i}^*(x)$ is the PDF of a $N\big(\bar{J}_b - \bar{J}_j, \frac{\sigma_b^2}{N_b+\Delta_b} + \frac{\sigma_j^2}{N_j}\big)$ random variable. If $i = j$, then $f_{b,j,i}^*(x)$ is the PDF of a $N\big(\bar{J}_b - \bar{J}_j, \frac{\sigma_b^2}{N_b} + \frac{\sigma_j^2}{N_j+\Delta_j}\big)$ random variable. If i is neither j nor b, then no new information is available to distinguish designs j and b, and $f_{b,j,i}^*(x) = f_{b,j}(x)$.

EEOC(i) is the estimated expected opportunity cost before additional Δ_i replications are applied for design i, for any i, and *EEOC*(i) can be computed easily. Our goal is to minimize the E[OC] of a potentially incorrect selection. Doing so optimally and in full generality is a challenging optimization problem with an unknown solution. Inspired by the greedy OCBA approach [Chen *et al.*, 1997; 1999], we reduce E[OC] by sequentially minimizing the upper bound in Equation (3.35), *EEOC*, for E[OC]. At each stage, we allocate additional samples in a manner that greedily reduces the *EEOC* as much as possible. That improvement is based upon, *EEOC*(i), which is a measure of the improvement in *EEOC* at each step. The effect of running an additional few replications on the expected opportunity cost, *EEOC*, is estimated by:

$$\begin{aligned} D_i &= EEOC(i) - EEOC \\ &= \int_0^{+\infty} x f^*_{b,i,i}(x)dx - \int_0^{+\infty} x f_{b,i}(x)dx \leq 0, \quad i \neq b, \qquad (3.38) \end{aligned}$$

and

$$\begin{aligned} D_b &= EEOC(b) - EEOC \\ &= \sum_{i=1, i\neq b}^{k} \left\{ \int_0^{+\infty} x f^*_{b,i,b}(x)\,dx - \int_0^{+\infty} x f_{b,i}(x)\,dx \right\} \leq 0, \quad (3.39) \end{aligned}$$

where D_i and D_b represent the reduction of *EEOC*(i) and *EEOC*(b), respectively, at each step. These inequalities can be verified with a bit of algebra. They imply that more information leads to smaller losses, on average.

Note that before conducting the simulation, neither the *EEOC* nor a good way to allocate the simulation budget is known. Therefore, all designs are initially simulated with n_0 replications in the first stage. We then sequentially allocate replications to the designs that provide the greatest reduction in the *EEOC*. Note the D_i and D_b are less than zero. This is consistent with our intuition that the expected opportunity cost will decrease as more samples are observed. With this approximation, the Δ_i, $i = 1, \ldots, k$, should not be too large. In summary, we have the following new selection procedure.

OCBA–EOC Procedure

```
INPUT       $k, T, \Delta, \tau, n_0$ ($T$-$kn_0$ is a multiple of $\Delta$ and $n_0 \geq 5$).
            Specifically, choose a number of designs per stage to
            simulate, $m$, and a (small) total number of simulation
            replications per stage, $\Delta$, such that $\tau = \Delta/m$ is an
            integer, and a total number of replications $T$ to run.
INITIALIZE  $l \leftarrow 0$;
            Perform $n_0$ simulation replications for all
            designs; $N_1^l = N_2^l = \cdots = N_k^l = n_0$.
LOOP        WHILE $\sum_{i=1}^{k} N_i^l < T$ DO
  UPDATE      Calculate sample means $\bar{J}_i = \frac{1}{N_i^l}\sum_{j=1}^{N_i^l} L(\theta_i, \omega_{ij})$, and sample standard deviation $s_i = \sqrt{\frac{1}{N_i^l-1}\sum_{j=1}^{N_i^l}(L(\theta_i, \omega_{ij}) - \bar{J}_i)^2}$, $i = 1, \ldots, k$,
              using the new simulation output; compute,
              $i = 1, \ldots, k$; find $b = \arg\min_i \bar{J}_i$; Calculate
              $D_i$, for $i = 1, \ldots, k$, using Equations (3.38)
              and (3.39).
  ALLOCATE    Find the set $S(m) \equiv \{i : D_i$ is among the $m$
              lowest values$\}$. Increase the computing budget by $\tau$ for design $i \in S(m)$, i.e., $N_i^{l+1} =$
              $N_i^l + \tau$ if $i \in S(m)$, $N_i^{l+1} = N_i^l$, otherwise.
  SIMULATE    Perform additional $\max(N_i^{l+1} - N_i^l, 0)$ simulations for design $i$, $i = 1, \ldots, k$; $l \leftarrow l + 1$.
END OF LOOP
```

Further details about this procedure can be found in He *et al.* [2007]. Since this selection procedure is derived based on the notion of OCBA, we named it OCBA–EOC to be consistent with the previous allocation procedure. The original OCBA is $P\{\text{CS}\}$ oriented, whereas the OCBA–EOC procedure in this section is focused on reducing E[OC]. As shown in He *et al.* [2007], the OCBA–EOC procedure compares quite favorably with several other procedures. Furthermore, it is also shown that the OCBA procedure presented in earlier sections performs very well not only for maximizing $P\{\text{CS}\}$, but also minimizing E[OC]. Therefore, for the purpose of minimizing E[OC], one can either use OCBA–EOC, or simply continue using the OCBA procedure.

3.7. OCBA Derivation Based on Classical Model

In this chapter, we have developed several OCBA schemes based on the Bayesian model which has some advantages in offering intuitive explanations of the methodology development and resulting allocation. This section presents the problem formulation and solution derivation using a classical (frequentist) model as an alternative. With the same objective of maximizing the probability of correct selection, we will show that the obtained OCBA solution using classical statistical model is identical to the one obtained using Bayesian model in Section 3.3.

We introduce the following notation. Some have been defined before; but we give all the necessary definitions here for purpose of completeness.

J_i: mean of design i,
σ_i^2: variance of design i,
$L(\theta_i, \omega_{ij})$: performance estimate obtained from the output of the j-th simulation replication for design i.
N_i: number of simulation replications for design i,
$\bar{J}_i$: sample mean of design i, i.e., $\bar{J}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} L(\theta_i, \omega_{ij})$,
$t = \arg\min_i J_i$, i.e., design t is the true best design,

$$\delta_{t,i} = J_t - J_i, \quad \text{and}$$
$$\sigma_{t,i}^2 = \frac{\sigma_t^2}{N_t} + \frac{\sigma_i^2}{N_i}.$$

As before, we assume that the simulation output is i.i.d. and has normal distribution with mean J_i and variance σ_i^2, i.e.,

$$L(\theta_i, \omega_{ij}) \sim N(J_i, \sigma_i^2). \tag{3.40}$$

In the classical model, the means and variances for all designs are assumed to be known, i.e., J_i, σ_i^2, and t are known in the derivation. The objective is to find a simulation budget allocation that maximizes the probability of correct selection ($P\{\text{CS}\}$), where "correct selection" is defined as picking the best design based on sample means from simulation output. Specifically, $P\{\text{CS}\}$ is defined as the probability that the true best design has the smallest sample mean and so can be selected correctly, which is

$$P\{\text{CS}\}_\text{C} = P\{\bar{J}_t < \bar{J}_i, \quad i \neq t\}. \tag{3.41}$$

In notation, a subscript "C" is added to distinguish it from the earlier $P\{\text{CS}\}$ definition based on Bayesian model. "C" stands for Classical model.

With normality and independence assumption,

$$\bar{J}_i \sim N\left(J_i, \frac{\sigma_i^2}{N_i}\right). \tag{3.42}$$

As N_i increases, the variance of $\bar{J}_i$ decreases, and so $P\{\text{CS}\}_\text{C}$ increases. The OCBA problem is how we should increase N_i in a way that $P\{\text{CS}\}_\text{C}$ can be maximized. More precisely, we wish to choose $N_1, N_2, \ldots, N_k$ such that $P\{\text{CS}\}_\text{C}$ is maximized, subject to a limited computing budget T,

$$\begin{aligned} &\max_{N_1,\ldots,N_k} \quad P\{\text{CS}\}_\text{C} \\ &\text{s.t. } N_1 + N_2 + \cdots + N_k = T. \end{aligned} \tag{3.43}$$

Since there is no simple close-form expression for $P\{\text{CS}\}_\text{C}$, we approximate it using the Bonferroni inequality, similar to the approximation *APCS-B* presented in Lemma 3.1.

$$\begin{aligned} P\{\text{CS}\}_\text{C} &= P\left\{\bigcap_{i=1,i\neq t}^{k} (\bar{J}_t - \bar{J}_i < 0)\right\} \\ &\geq 1 - \sum_{i=1,i\neq t}^{k} [1 - P\{\bar{J}_t - \bar{J}_i < 0\}] \\ &= 1 - \sum_{i=1,i\neq t}^{k} P\{\bar{J}_t > \bar{J}_i\} \equiv \mathit{APCS\text{-}C}. \end{aligned}$$

$P\{\mathrm{CS}\}_{\mathrm{C}}$ is approximated by*APCS-C*. We now consider the following problem as an approximation to Equation (3.43).

$$\max_{N_1,\ldots,N_k} 1 - \sum_{i=1,i\neq t}^{k} P\{\bar{J}_t > \bar{J}_i\}$$

$$\text{s.t.} \sum_{i=1}^{k} N_i = T. \tag{3.44}$$

For the objective function,

$$\begin{aligned}\sum_{i=1,i\neq b}^{k} P\{\bar{J}_t > \bar{J}_i\} &= \sum_{\substack{i=1\\i\neq t}}^{k} \int_0^{\infty} \frac{1}{\sqrt{2\pi}\sigma_{t,i}} e^{-\frac{(x-\delta_{t,i})^2}{2\sigma_{t,i}^2}} dx \\ &= \sum_{\substack{i=1\\i\neq t}}^{k} \int_{-\frac{\delta_{t,i}}{\sigma_{t,i}}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy,\end{aligned}$$

where

$$\sigma_{t,i}^2 = \frac{\sigma_t^2}{N_t} + \frac{\sigma_i^2}{N_i}.$$

Let F be the Lagrangian relaxation of Equation (3.44):

$$F = 1 - \sum_{\substack{i=1\\i\neq t}}^{k} \int_{-\frac{\delta_{t,i}}{\sigma_{t,i}}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy - \lambda \left(\sum_{i=1}^{k} N_i - T \right), \tag{3.45}$$

which is identical to the Lagrangian in Equation (3.13) except that b is replaced by t. To find the stationary point of the Lagrangian function, we set $\frac{\partial F}{\partial N_i} = 0$ for each value of i. Since the Lagrangian functions are basically identical, following the same procedure in Section 3.3, we can obtain a similar local optimal solution summarized in the following theorem.

Theorem 3.5. *Given a total number of simulation replications T to be allocated to k competing designs with means $J_1, J_2, \ldots, J_k$, and finite variances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_k^2$, respectively, as $T \to \infty$, the* Approximate Probability of Correct Selection (APCS-C) *can be* *asymptotically maximized when*

$$\frac{N_i}{N_j} = \left(\frac{\sigma_i/\delta_{t,i}}{\sigma_j/\delta_{t,j}}\right)^2, \quad i,j \in \{1,2,\ldots,k\}, \quad \text{and} \quad i \neq j \neq t,$$

$$N_t = \sigma_t \sqrt{\sum_{i=1, i\neq t}^{k} \frac{N_i^2}{\sigma_i^2}}.$$

□

We assume the means J_i and variances σ_i^2 are known when deriving Theorem 3.5. In practice, as means and variances are unknown, we employ a heuristic approach to tackle these problems. A sequential procedure is used to approximately estimate these means and variances, and also t using sample statistics. Each design is initially simulated with n_0 replications in the first stage, and additional replications are allocated to individual solutions incrementally from Δ replications to be allocated in each subsequent stage until the simulation budget T is exhausted. When $T \to \infty$, since every solution has been allocated with simulation runs infinitely often, we can show that the allocation rule using the sample average and sample standard deviation will converge to the rule using true means and true variances. Even though these sample statistics and t are varying over iterations, the impact of these approximations will decay asymptotically. In summary, we have the following heuristic algorithm which is actually the same as the procedure given in Section 3.3 except that b is replaced by t.

OCBA Procedure (Classical Model)

```
INPUT        $k, T, \Delta, n_0$ ($T$-$kn_0$ is a multiple of $\Delta$ and $n_0 \geq 5$);
INITIALIZE   $l \leftarrow 0$;
             Perform $n_0$ simulation replications for all designs; $N_1^l = N_2^l = \cdots = N_k^l = n_0$.
LOOP         WHILE $\sum_{i=1}^{k} N_i^l < T$ DO
  UPDATE     Calculate sample means $\bar{J}_i = \frac{1}{N_i^l}\sum_{j=1}^{N_i^l} L(\theta_i, \omega_{ij})$, and sample standard deviation $s_i = \sqrt{\frac{1}{N_i^l - 1}\sum_{j=1}^{N_i^l}(L(\theta_i, \omega_{ij}) - \bar{J}_i)^2}$, $i = 1, \ldots, k$,
                using the new simulation output; compute,
                $i = 1, \ldots, k$; find $t = \arg\min_i \bar{J}_i$.
  ALLOCATE      Increase the computing budget by $\Delta$ and
                calculate the new budget allocation, $N_1^{l+1}$,
                $N_2^{l+1}, \ldots, N_k^{l+1}$, according to
                (1) $\frac{N_i^{l+1}}{N_j^{l+1}} = \left(\frac{s_i(\bar{J}_t - \bar{J}_j)}{s_j(\bar{J}_t - \bar{J}_i)}\right)^2$, for all $i \neq j \neq b$,
                and
                (2) $N_t^{l+1} = s_t \sqrt{\sum_{i=1, i \neq t}^{k} \left(\frac{N_i^{l+1}}{s_i}\right)^2}$,
  SIMULATE      Perform additional $\max(N_i^{l+1} - N_i^l, 0)$ simu-
                lations for design $i$, $i = 1, \ldots, k$; $l \leftarrow l + 1$.
END OF LOOP
```

In conclusion, we basically obtain the same asymptotical solution and OCBA procedure, no matter whether Bayesian or classical models are applied.

Chapter 4

Numerical Implementation and Experiments

With the development of standard OCBA procedures in Chapter 3, we turn our attention to practical implementation issues. First we will test the OCBA procedure and compare it with several well known procedures empirically. Numerical results show that OCBA outperforms all compared procedures. Secondly we will extensively discuss how the parameters in the OCBA procedures should be determined. Sensitivity of the simulation performance to those parameters will be discussed. Finally, some additional implementation issues, such as rounding the number of simulation replications into integers, will be discussed. In Appendix D, we will provide a C/C++ code of the OCBA algorithm.

4.1. Numerical Testing

In this section we test the OCBA algorithm and compare it with several different allocation procedures by performing a series of numerical experiments.

4.1.1. *OCBA algorithm*

The OCBA procedure has already been extensively discussed in Chapter 3. We briefly summarize it as follows.

OCBA Procedure (Maximizing $P\{\text{CS}\}$ or Minimizing Simulation Cost)

```
INPUT           $k$, $T$, $\Delta$, $n_0$ ($T - kn_0$ a multiple of $\Delta$ and $n_0 \geq 5$);
INITIALIZE      $l \leftarrow 0$;
                Perform $n_0$ simulation replications for all designs; $N_1^l = N_2^l = \cdots = N_k^l = n_0$.
LOOP            WHILE $\sum_{i=1}^{k} N_i^l < T$ (or $APCS < P^*$) DO
    UPDATE      Calculate sample means $\bar{J}_i = \frac{1}{N_i^l}\sum_{j=1}^{N_i^l} L(\theta_i, \omega_{ij})$, and sample standard deviation $s_i = \sqrt{\frac{1}{N_i^l - 1}\sum_{j=1}^{N_i^l}(L(\theta_i, \omega_{ij}) - \bar{J}_i)^2}$, $i = 1, \ldots, k$, using the new simulation output; find $b = \arg\min_i \bar{J}_i$.
    ALLOCATE    Increase the computing budget by $\Delta$ and calculate the new budget allocation, $N_1^{l+1}$, $N_2^{l+1}, \ldots, N_k^{l+1}$, according to
                (1) $\frac{N_i^{l+1}}{N_j^{l+1}} = \left(\frac{s_i(\bar{J}_b - \bar{J}_j)}{s_j(\bar{J}_b - \bar{J}_i)}\right)^2$, for all $i \neq j \neq b$, and
                (2) $N_b^{l+1} = s_b\sqrt{\sum_{i=1, i\neq b}^{k}\left(\frac{N_i^{l+1}}{s_i}\right)^2}$,
    SIMULATE    Perform additional $\max(N_i^{l+1} - N_i^l, 0)$ simulations for design $i$, $i = 1, \ldots, k$; $l \leftarrow l + 1$.
END OF LOOP
```

Note that if *APCS* is used as a stopping criterion, it can be estimated by

$$APCS = 1 - \sum_{i=1, i\neq b}^{k} \Phi\left(\frac{\bar{J}_b - \bar{J}_i}{\sqrt{\frac{s_b^2}{N_b^l} + \frac{s_i^2}{N_i^l}}}\right).$$

using updated simulation output.

Remark 4.1. The resulting N_i in the ALLOCATE step is a continuous number that must be rounded to an integer. In the numerical experiments in the next section, N_i is rounded to the nearest integer such that the summation of additional simulation replications for all solutions equals Δ. Note that there may not always exist a solution that satisfies all three constraints. It actually occurs when at least one solution has been over simulated, i.e., $N_i^{l+1} < N_i^l$. In this case, we have to relax the constraint. For ease of control of the simulation experiment, we can choose to maintain the constraint $\sum_{i=1}^k N_i^{l+1} = T^{l+1}$ and apply some heuristics to round N_i^{l+1} for all i to nearest integers. We have found numerically that the performance is not sensitive to how we round N_i, probably due to the robustness of a sequential procedure.

Alternative simpler OCBA procedure

When the simulation is relatively expensive, the computation cost of the ALLOCATION step becomes negligible. In this case, Δ should be as small as possible and can be set as 1. To avoid the numerical error occurred when rounding N_i^{l+1} into an integer, we suggest a numerically robust alternative. The ALLOCATE and SIMULATE steps are revised as follows.

ALLOCATE Increase the computing budget by one and calculate the new budget allocation, N_1^{l+1}, $N_2^{l+1}, \ldots, N_k^{l+1}$, according to

(1) $\frac{N_i^{l+1}}{N_j^{l+1}} = \left(\frac{s_i(\bar{J}_b - \bar{J}_j)}{s_j(\bar{J}_b - \bar{J}_i)}\right)^2$, for all $i \neq j \neq b$, and

(2) $N_b^{l+1} = s_b \sqrt{\sum_{i=1, i\neq b}^k \left(\frac{N_i^{l+1}}{s_i}\right)^2}$,

leave N_i^{l+1} as a decimal number and find $i^* = \arg\max_i(N_i^{l+1} - N_i^l)$.

SIMULATE Perform additional one simulation for design i^*; $N_{i*}^{l+1} = N_{i*}^l + 1$; $N_i^{l+1} = N_i^l$, for $i \neq i^*$; $l \leftarrow l + 1$.

Intuitively, we determine which design is the most starving one in terms of the need of additional simulation, and then simulate that design for one additional replication. This procedure is iteratively continued until the total budget T is exhausted or the estimated *APCS* is sufficiently high. As shown in Section 4.2, this simpler procedure performs equally well in our numerical testing.

4.1.2. *Different allocation procedures for comparison*

In addition to the OCBA algorithm, we test several procedures and compare their performances. Among them, equal allocation which simulates all design alternatives equally; the two-stage procedures of Dudewicz and Dalal [1975] and Rinott [1977] that allocate replications proportional to the estimated variances (and which is highly popular in simulation literature); and Proportional To Variance (PTV) which is a sequential allocation procedure modified from the Rinott procedure. We briefly summarize the compared allocation procedures as follows.

Equal allocation

This is the simplest way to conduct simulation experiments and has been widely applied. The simulation budget is equally allocated to all designs, that is, $N_i = T/k$ for each i. The performance of equal allocation will serve as a benchmark for comparison.

Well-known two-stage procedures

The two-stage procedures of Dudewicz and Dalal [1975] and Rinott [1977] and their variations have been used widely in simulation applications. See Bechhofer *et al.* [1995] for a systematic discussion of two-stage procedures and their recent development. In this section, we compare the performance of these two-stage procedures with that of the OCBA algorithm through numerical examples. We also compare the computation effort required to achieve a same $P\{\text{CS}\}$.

Unlike the OCBA approach, two-stage procedures are based on the classical ranking-and-selection model. It is convenient to define the indices such that $J_{[1]} \leq J_{[2]} \leq \cdots \leq J_{[k]}$. As in OCBA, the goal in two-stage procedures is also to select a design with the smallest mean, $J_{[1]}$. Based on the "indifference zone" idea, one may be willing to choose design 2 if $J_{[1]}$ and $J_{[2]}$ are very close (i.e., $J_{[2]} - J_{[1]} < d^*$, where d^* is the indifference zone parameter). By doing so, the procedure avoids conducting a large number of simulation replications to distinguish between small differences. More specifically, we intend to determine the number of simulation replications to ensure

$$P\{\bar{J}_{[1]} < \bar{J}_{[i]}, \forall i \neq 1\} \geq P^*$$
$$\text{whenever } J_{[2]} - J_{[1]} < d^*,$$

where P^* is the confidence level requirement. The procedure is as follows.

Before performing the simulation experiment, we specify three parameters: P^*, n_0, and d^*. Let h be the constant that solves a certain integral equation guaranteeing the validity of Rinott or Dudewicz and Dalal procedures given P^*, n_0, and k (e.g., h can be obtained from the tables in Wilcox [1984] for Rinott's procedure, and from the tables in Gibbons *et al.* [1977] for Dudewicz and Dalal's procedure). In the first stage, all designs are simulated for n_0 replications. Based on the sample standard deviation estimate (s_i) obtained from the first stage and given the desired correct selection probability P^*, the number of additional simulation replications for each design in the second stage is determined by:

$$N_i = \max(n_0, \lceil (hs_i/d^*)^2 \rceil), \quad \text{for } i = 1, 2, \ldots, k, \tag{4.1}$$

where $\lceil \bullet \rceil$ is the integer "round-up" function.

In short, the computing budget is allocated proportionally to the estimated sample variances. The major drawback is that only the information on variances is used when determining the simulation allocation, while the OCBA algorithm utilizes the information on both means and variances. Further it is assumed a least favorable configuration when deriving the allocation, which is more like a worst case analysis and so is very conservative. As a result, the performance of two-stage procedures is not as good as others. We do, however, include it in some of our testing due to its popularity in simulation literature.

Proportional to variance (PTV)

This is a sequential modified version of the two-stage Rinott procedure, based on the observation that Equation (4.1) implies that N_i is proportional to the estimated sample variance s_i^2. Thus, the PTV procedure sequentially determines $\{N_i\}$ based on the newly updated sample variances by replacing the **ALLOCATE** step of the OCBA algorithm by

ALLOCATE Increase the computing budget by Δ and calculate the new budget allocation, N_1^{l+1}, $N_2^{l+1}, \ldots, N_k^{l+1}$, according to

$$\frac{N_1^{l+1}}{s_1^2} = \frac{N_2^{l+1}}{s_2^2} = \cdots = \frac{N_k^{l+1}}{s_k^2}.$$

Thus the number of replications for each design grows in proportion to the sample variance. Note that the indifference-zone parameter has been removed in this modification in order to make it comparable to the other procedures.

While we do not claim that the $P\{\text{CS}\}$ guarantee is maintained, this approach maintains the spirit of allocating replications proportional to the sample variance.

4.1.3. *Numerical experiments*

The numerical experiments starts with a test on the worker allocation problem (Example 4.1) followed by a series of generic tests. In all the numerical illustrations, we estimate $P\{\text{CS}\}$ by counting the number of times we successfully find the true best design out of 100,000 independent applications of each selection procedure. $P\{\text{CS}\}$ is then obtained by dividing this number by 100,000, representing the correct selection frequency.

Each of the procedures simulates each of the k designs for $n_0 = 10$ replications initially. Dudewicz and Dalal's and Rinott's procedures allocate additional replications in a second stage (so the total number is not fixed *a priori*), whereas the other procedures allocate replications incrementally by $\Delta = 20$ each time until the total budget, T, is consumed. For each level of computing budget, we estimate the achieved $P\{\text{CS}\}$. We anticipate that all procedures will obtain a higher $P\{\text{CS}\}$ as the computing budget increases.

Since Dudewicz and Dalal's and Rinott's procedures are two-stage indifference-zone procedures, we must specify the values for the desired probability of correct selection, P^*, and the indifference zone d to satisfy the condition that $J_{[2]} - J_{[1]} \geq d$, where a smaller d implies a higher required computation cost based on Equation (4.1). In practice, the value of $J_{[2]}$ or $J_{[1]}$ is unknown beforehand, but for benchmarking purposes, we set $d = J_{[2]} - J_{[1]}$, which leads to the *minimum* computational requirement (or maximum efficiency) for these procedures. As is done for the other procedures, the resulting $P\{\text{CS}\}$ can be estimated over the 100,000 independent experiments. Since the required computation cost varies from one experiment to another, we will indicate the average number of total replications based on the 100,000 independent experiments.

Experiment 4.1. Worker allocation problem

This numerical test is based on the worker allocation problem presented in Chapter 1, Example 4.1. In summary, this problem represents the challenge of allocating 31 workers within a two-stage queue where each stage of the queue can contain no less than 11 workers as shown in Figure 4.1.

Denote C_1 and C_2 as the numbers of workers allocated to nodes 1 and 2. Thus $C_1 + C_2 = 31$, $C_1 \geq 11$, and $C_2 \geq 11$. There are

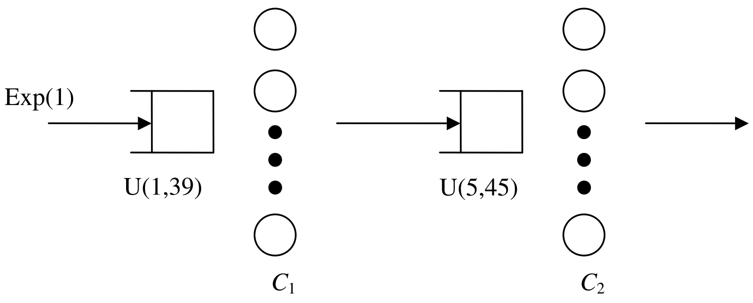


Figure 4.1. A two-stage queuing system where both C_1, and C_2 must be greater than 10.

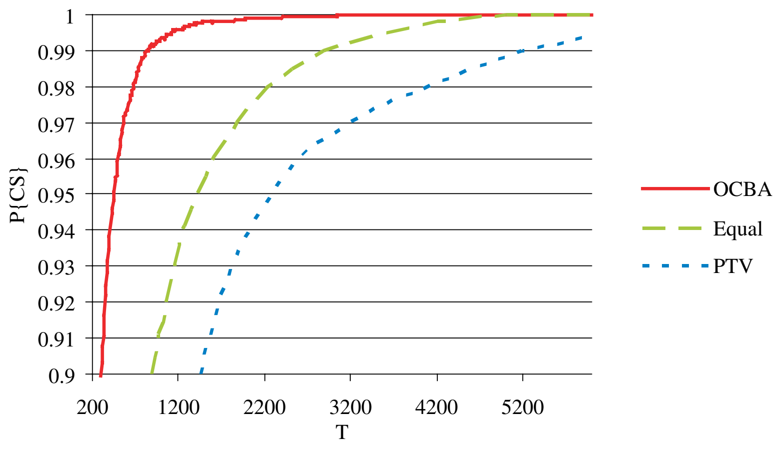


Figure 4.2. $P\{\text{CS}\}$ versus T using three sequential allocation procedures for Experiment 4.1.

10 alternative combinations of (C_1, C_2). We want to choose the best alternative of (C_1, C_2) so that the average system time for the first 100 customers is minimized. Since there is no closed-form analytical solution for the estimation of the system time, stochastic (discrete-event) simulation can be performed to find the best design.

To characterize the performance of different procedures as a function of T, we vary T between 200 and 8000 for all of the sequential procedures and the estimated achieved $P\{\text{CS}\}$ as a function of T is shown in Figure 4.2. We see that all procedures obtain a higher $P\{\text{CS}\}$ as the available computing budget increases. However, OCBA achieves a same $P\{\text{CS}\}$ using the lowest amount of computing budget. Table 4.1 shows the computation costs to attain $P\{\text{CS}\} = 0.95$ and 0.99 for OCBA, Equal, and PTV, respectively.

Table 4.1. The computation costs to attain $P\{\text{CS}\} = 0.95$ or 0.99 using different sequential procedures.

$P\{\text{CS}\}$	OCBA	Equal	PTV
0.95	470	1,450	2,270
0.99	850	2,890	5,230

It is worth noting that PTV (a sequential version of Rinott's procedure) performs worse than simple equal allocation for this problem. This performance is because PTV determines the number of simulation samples for all designs using only the information of sample variances. For this particular worker allocation problem, the best designs have the lowest variance and the worst designs have the largest variance. As a result, PTV generally allocates less time to designs that perform well, thus achieving lower P{CS} for the same T values.

Overall, both Equal and PTV are much slower than OCBA. Figure 4.3 shows the average total number of simulation replications allocated, N_i, for all i, to achieve a $P\{\text{CS}\}$ of 0.99% using OCBA, PTV, and the Equal procedure, respectively. It is noteworthy to compare the total number of simulation replications to the actual average system times and variances (as determined by simulating each design 100,000 times) shown in Figure 4.4. Note that the budget allocation by OCBA is quite different from that using PTV. PTV determines the number of simulation replications for all designs using only the

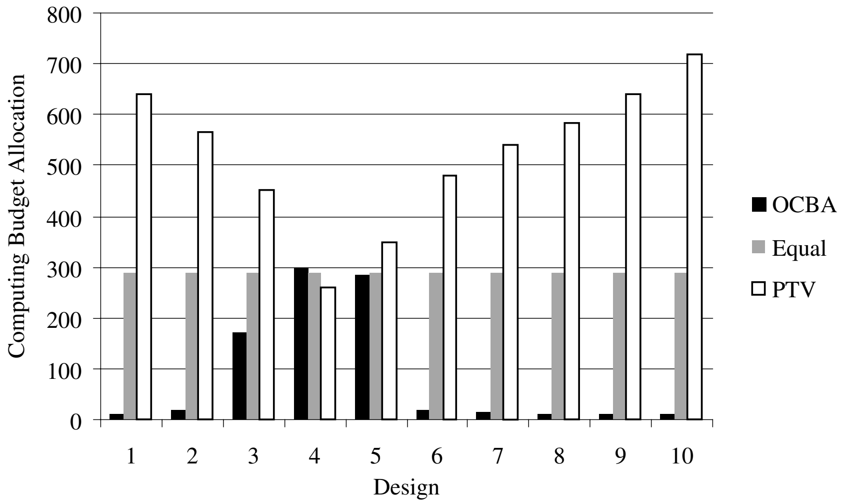


Figure 4.3. Computing budget allocation as determined by the OCBA, Equal, and PTV procedure for the worker allocation problem to achieve a $P\{\text{CS}\}$ of 0.99%.

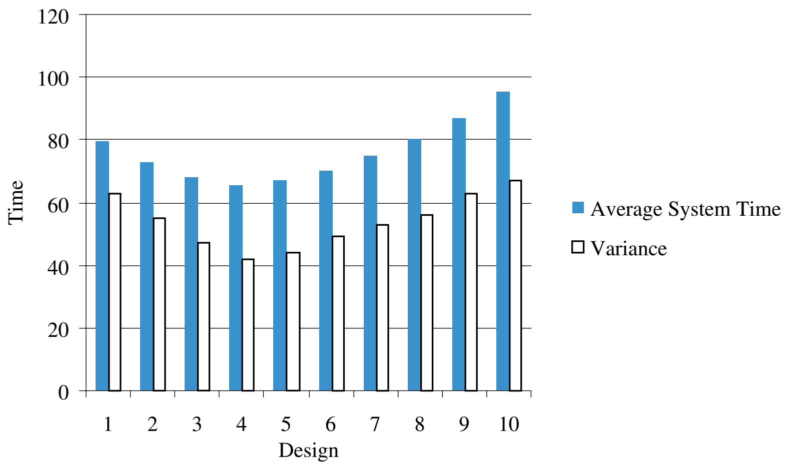
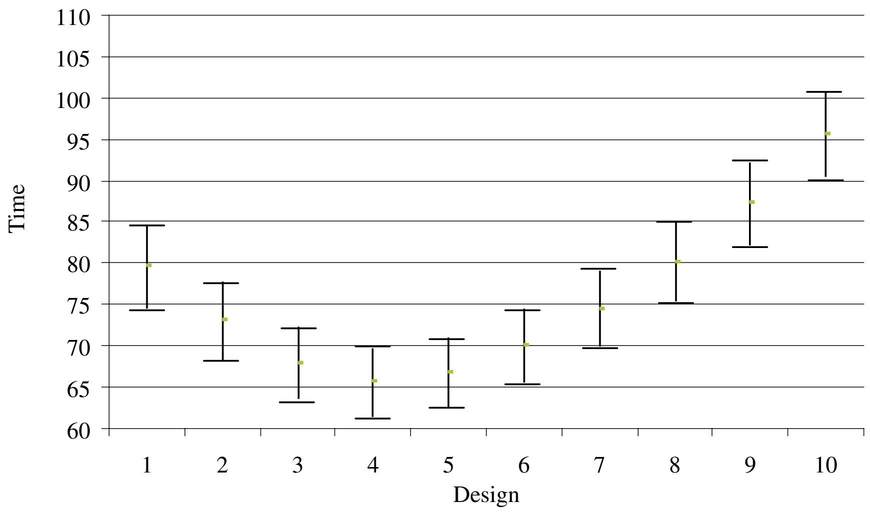


Figure 4.4. Actual average system time and variance for each design.

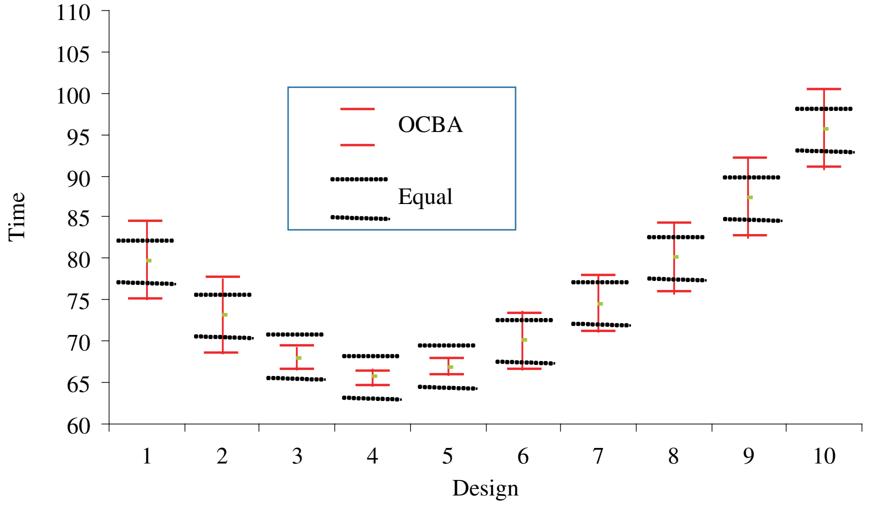
information on sample variances. On the other hand, OCBA exploits the information on both sample means and variances, achieving a much more efficient performance. It is worthy to note that OCBA simulates the top three designs (designs 3, 4, and 5) much more than all of the other designs combined whereas PTV simulates these designs less than other designs because they have smaller variance.

It is also useful to look at the confidence intervals at various T values to better understand how OCBA allocates available computing time across the 10 designs. Figures 4.5(a) and 4.5(b) show the Equal and OCBA 95% confidence intervals for $T = 100$ (after n_0 replications of each design) and $T = 500$, respectively. Figure 4.5(b) shows that Equal Allocation reduced the width of confidence intervals for all designs, while OCBA concentrates on the critical designs. As a result, OCBA can better determine whether the observed best design is indeed the true best design or not.

Finally, we test Dudewicz and Dalal's and Rinott's procedures for $P^* = 0.75$, $P^* = 0.90$, and $P^* = 0.95$. As Table 4.2 shows, both of these approaches are extremely conservative (e.g., obtaining 96.1% of actual $P\{\text{CS}\}$ when the desired setting is only 75%) and result in much higher computation cost to achieve a same $P\{\text{CS}\}$ as compared with OCBA.



(a)



(b)

Figure 4.5. (a) 95% confidence interval for $T = 100$ (after n_0 replications of each design). Since OCBA is simulated equally in the first stage, the OCBA and Equal confidence intervals are the same. (b) 95% confidence intervals for $T = 500$ for equal allocation and OCBA.

Table 4.2. Results for Dudewicz and Dalal's and Rinott's procedure.

	Dudewicz and Dalal		Rinott	
P^*	Avg Total # of Replications	$P\{\text{CS}\}$	Avg Total # of Replications	$P\{\text{CS}\}$
75%	2649	96.1%	3191	97.0%
90%	5042	98.7%	5264	99.1%
95%	6697	99.6%	6889	99.6%

Experiment 4.2. Simple generic examples

The second experiment is a series of simple generic tests. The purpose is to compare how different procedures perform under different well controlled settings. Since the performances of Dudewicz and Dalal's and Rinott's procedures basically follow that of PTV, but their required computing budget is far beyond the range we are considering here, we exclude those two-stage procedures from the numerical testing here.

Experiment 4.2.1. Simulation output is normally distributed

We assume there are ten design alternatives whose simulation output is normally distributed. Suppose $L(\theta_i, \omega) \sim N(i, 6^2)$, $i = 1, 2, \cdots, 10$. We want to find a design with the minimum mean. It is obvious that design 1 is the actual best design. The information of $N(i, 6^2)$ is solely used to generate simulation output, but not used in determining the simulation budget allocation or selecting the best design.

Figure 4.6 shows the test results using OCBA, Equal, and PTV procedures discussed in Sections 4.1.1 and 4.1.2 (only $P\{\text{CS}\}$ values greater than or equal to 0.70 are shown here). We see that all procedures obtain a higher $P\{\text{CS}\}$ as the available computing budget increases. However, OCBA achieves the same $P\{\text{CS}\}$ with a lower amount of computing budget than other procedures. In particular, we indicate the computation costs in order to attain $P\{\text{CS}\} = 99\%$ for different procedures in Table 4.3, showing that OCBA can reduce the simulation time by about 75% (ratio of 4 to 1) for $P\{\text{CS}\} = 99\%$.

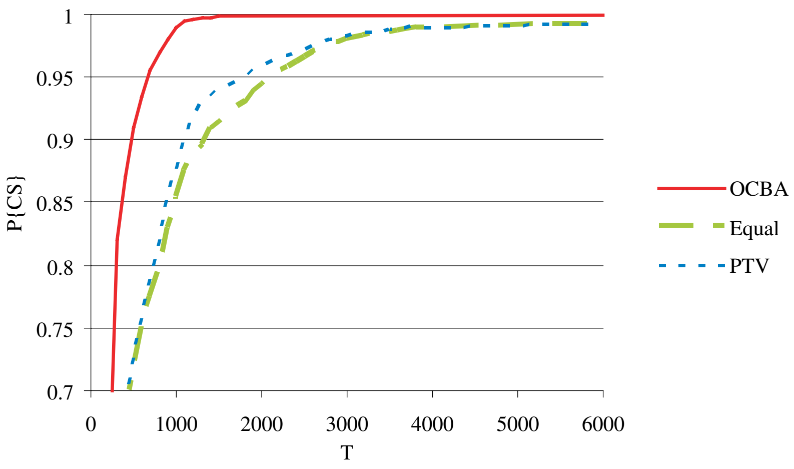


Figure 4.6. $P\{\text{CS}\}$ versus T using three sequential allocation procedures for Experiment 4.2.1.

Table 4.3. Computation cost required to obtain P{CS} = 0.99.

P{CS}	OCBA	Equal	PTV
0.99	1,100	4,400	4,000

Experiment 4.2.2. Normal distribution with larger variance

This is a variant of Experiment 4.2.1. All settings are preserved except that the variance of each design is doubled. Namely, $L(\theta_i, \omega) \sim N(i, 2 \cdot 6^2)$, $i = 1, 2, \ldots, 10$. Figure 4.7 contains the simulation results for different allocation procedures (for $P\{\text{CS}\}$ greater than or equal to 0.70) and Table 4.4 contains the amount of computation time required to achieve a $P\{\text{CS}\} = 0.99$. We can see that the relative performances of different procedures are very similar to the previous experiment, except that bigger computing budgets are needed in order to obtain the same $P\{\text{CS}\}$ (due to larger variance). We also observe that as the design variance increases (doubles in this case) the relative "speed-up factor" of OCBA remains about the same (about four times in this case).

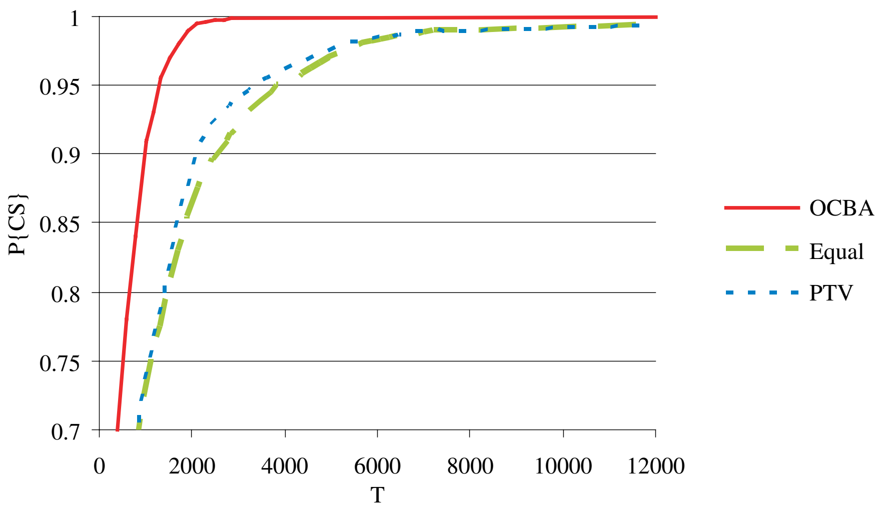


Figure 4.7. $P\{\mathrm{CS}\}$ versus T using three sequential allocation procedures for Experiment 4.2.2.

Table 4.4. Computation cost to achieve P{CS} of 0.99.

P{CS}	OCBA	Equal	PTV
0.99	2,100	8,500	7,500

Experiment 4.2.3. Simulation output is uniformly distributed

In this case, we consider a non-normal distribution for the performance measure: $L(\theta_i, \omega) \sim \mathrm{Uniform}(i-10.5, i+10.5)$, $i = 1, 2, \ldots, 10$. The endpoints of the uniform distribution are chosen such that the corresponding variance is close to that in Experiment 4.2.1. Again, we want to find a design with the minimum mean; design 1 is therefore the actual best design. All other settings are identical to Experiment 4.2.1. Figure 4.8 contains the simulation results for the three allocation procedures. We can see that the relative performances of the different procedures are very similar to previous results (i.e., OCBA performs three-times faster than equal allocation). The specific number of simulation iterations required to achieve a P{CS} of 0.99 is shown in Table 4.5.

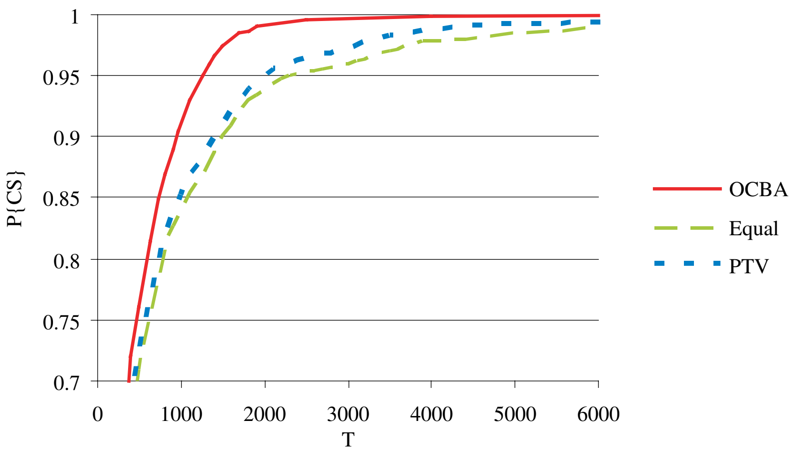


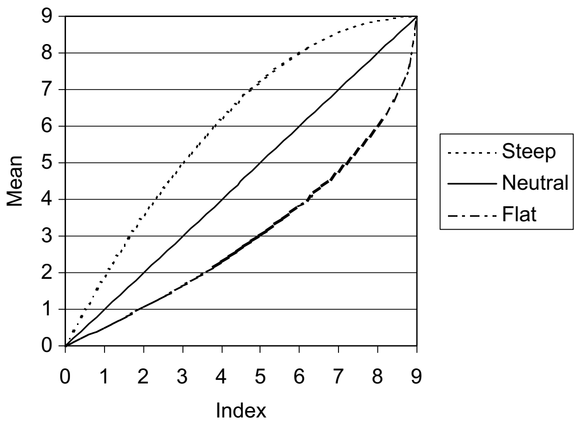
Figure 4.8. $P\{\text{CS}\}$ versus T using different allocation procedures for Experiment 4.2.3. The computation costs in order to attain $P\{CS\} = 99\%$ are indicated.

Table 4.5. Computation cost to achieve P{CS} of 0.99.

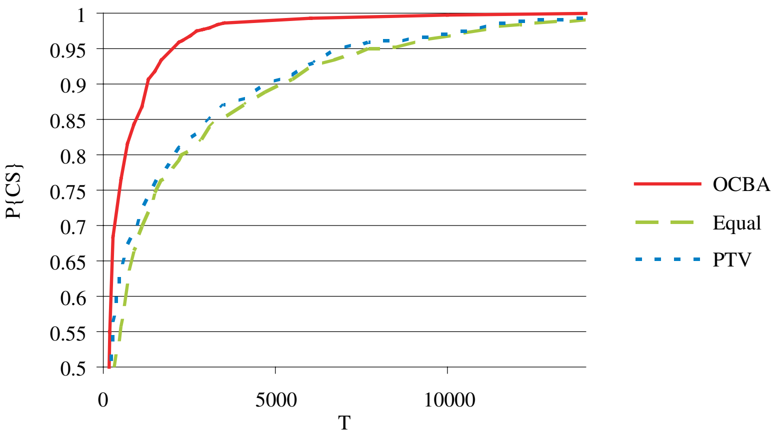
P{CS}	OCBA	Equal	PTV
0.99	1,900	6,000	4,400

Experiment 4.2.4. Flat and steep case

This test is another variant of Experiment 4.2.1. We consider three generic cases illustrated in Figure 4.9(a) (also shown in Ho *et al.* 1992): neutral, flat, and steep. The neutral case is already presented in Experiment 4.2.1. In the flat case, $L(\theta_i, \omega) \sim N(9 - 3\sqrt{9-i}, 6^2)$, $i = 1, 2, \ldots, 10$; and in the steep case $L(\theta_i, \omega) \sim N\big(9 - \left(\frac{9-i}{3}\right)^2, 6^2\big)$, $i = 1, 2, \ldots, 10$. In the flat case, good designs are closer; a larger computing budget is therefore needed to identify the best design given the same simulation estimation noise. On the other hand, it is easier to correctly select the best design in the steep case since the good designs are further spread out. The results of these predictions are confirmed in Figures. 4.9(b) and 4.9(c), and Table 4.6. As shown in Tables 4.3 (neutral case) and 4.6, the OCBA "speed-up factor" remains about the same for all the three cases.



(a)

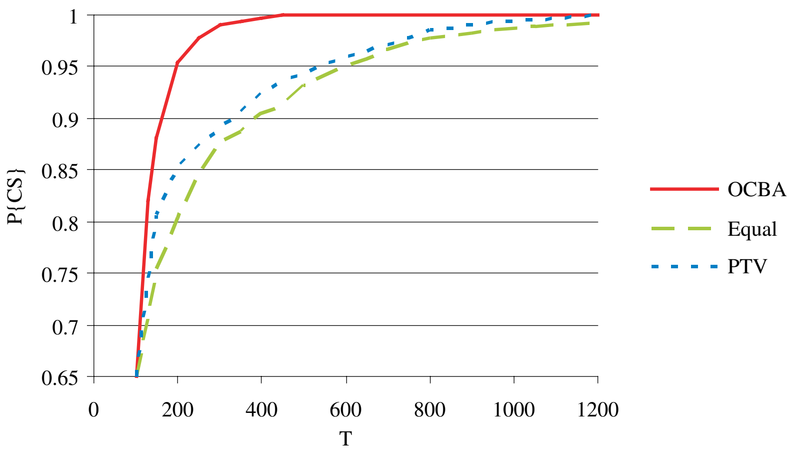


(b)

Figure 4.9. (a) Illustration of three generic cases: neutral, flat, steep. (b) $P\{\text{CS}\}$ versus T using three different allocation procedures for the flat case in Experiment 4.2.4. (c) $P\{\text{CS}\}$ versus T using three different allocation procedures for the steep case in Experiment 4.2.4.

Experiment 4.3. Larger design space

This experiment is a variant of Experiment 4.1. In this case, we study how well OCBA performs for larger numbers of design alternatives. In this case, instead of providing only 31 workers, we increase the number of workers up to 121 workers where there must be at



(c)

Figure 4.9. (*Continued*)

Table 4.6. Computation cost required to achieve a $P\{\text{CS}\}$ of 0.99 for the flat and steep case.

	P{CS}	OCBA	Equal	PTV
Flat Case	0.99	4,100	15,100	13,300
Steep Case	0.99	300	1,100	1,000

least 11 workers at each station. As a result, the number of possible design alternatives varies from 10 to 100. Table 4.7 lists some of the possibilities. In addition, we change the first station service time to U(20, 58). The second station service time remains the same. All other parameters and constraints of the original problem also remain the same.

In this test, we compare OCBA and equal allocation, and focus on the "speed-up factors" under OCBA. For both procedures, we record the minimum computation cost to reach $P\{\text{CS}\} = 99\%$: T_{OCBA} and T_{EA}. The "speed-up factor" using OCAB is given by the ratio $T_{\text{EA}}/T_{\text{OCBA}}$. It is also useful to measure the so-called *Equivalent Number of Alternatives with a Fixed Computing Budget*, ENAFCB(k), which is defined as the number of alternatives that can

Table 4.7. Number of maximum allowable workers to simulate varying numbers of designs.

Number of designs	Maximum number of workers
10	31
20	41
30	51
40	61
50	71
60	81
70	91
80	101
90	111
100	121

be simulated under the equal allocation procedure using the computing budget that is needed for OCBA to simulate k alternatives for reaching $P\{\text{CS}\} = 99\%$. For example, in the case of 10 alternatives, OCBA is 3.40 times faster than equal allocation (requires a computation cost of 850 instead of 2890 to reach a P{CS} of 0.99). Thus, OCBA can simulate 10 alternatives in the same time that equal allocation can simulate only $10/3.40 = 2.94$ designs. In this case, ENAFCB(10) for OCBA is 2.94. An alternative interpretation of this statement is that, under OCBA, we can simulate 10 alternative designs with only the equivalent effort of 2.94 designs.

Based on the approach described above, the speed-up factors and ENAFCB factors are shown below in Table 4.8. We see that OCBA is even more efficient as the number of designs increases. The higher efficiency is obtained because a larger design space gives the OCBA algorithm more flexibility in allocating the computing budget. In particular, ENAFCB(50) $=$ 3.94 means that with an equivalent effort of less than 4 alternatives, OCBA can simulate 50 design alternatives for Experiment 4.1. This advantage demonstrates that OCBA can provide the ability to simulate many more designs for limited available computational time (further enhancing the probability of identifying the correct design).

Table 4.8. The speed-up factor of using OCBA as compared with the use of equal allocation for Experiment 4.3.

Number of designs, k	5	10	25	50	75	100
Speed-up factor using OCBA	2.08	3.40	7.86	12.69	16.50	20.05
ENAFCB(k)	2.40	2.94	3.18	3.94	4.55	4.99

To thoroughly examine the speed-up property of OCBA, we further increase the number of designs in the testing. We extend the test by increasing the number of design alternatives from 100 to 1000 using an identical approach to that described in Table 4.7 (e.g., letting the maximum number of servers equal 21 more than the number of desired designs). Figure 4.10 shows that the speed-up factor initially increases almost linearly when the number of alternatives is small. However, when the number of alternatives is large, the speed-up converges to some maximum level. This is due to the initial sampling cost for OCBA. Initial sampling is needed to gather the first estimates for means and variances so that the optimal computing budget allocation can be determined. Because we do not assume any prior knowledge about the topology of the design space, this initial

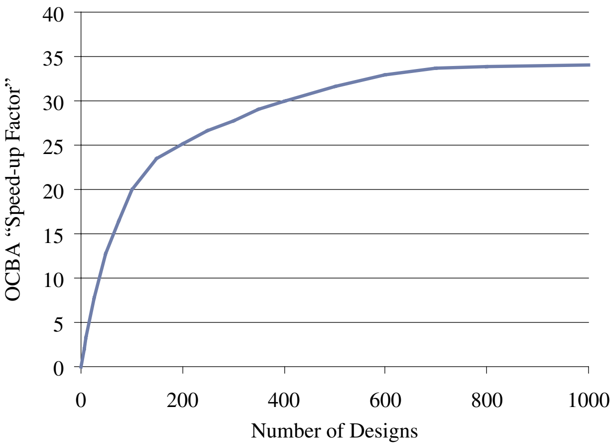


Figure 4.10. Speed-up factor versus number of designs using OCBA.

sampling cost is proportional to the number of alternatives and will become dominating when the number of alternatives is large.

4.2. Parameter Setting and Implementation of the OCBA Procedure

To apply the OCBA algorithms, we need to specify appropriate values for the algorithm parameters, n_0, the initial number of simulation replications, and Δ, the one-time computing budget increment. These two parameters may affect the performance of the algorithms. In this section, we provide guidelines for selecting these two parameters, although a good choice might be problem-specific. In addition, we offer some discussions about the approximations made in the OCBA procedures.

4.2.1. *Initial number of simulation replications, n_0*

It is well understood that n_0 cannot be too small as the estimates of the mean and the variance may be very poor, resulting in poor computing budget allocation. Nevertheless, a small n_0 gives us added flexibility for better allocation of the computing budget. On the other hand, if n_0 is too large, we may waste computation time in simulating non-promising designs (similar impact to having a large number of designs such as in Figure 4.10 so that the initial simulations become the dominant factor in algorithm efficiency). Intuitively, the effect of n_0 should be less significant when the total computing budget, T, is large.

We use the worker allocation example (Experiment 4.1) to illustrate the impact of different values of n_0. Figure 4.11 shows the probability of correct selection as a function of the total computing budget, T, for different n_0 values. When T is very small, low n_0 values achieve higher P{CS} values. However, from Figure 4.11, we observe that the achieved P\{CS\} appears to be fairly insensitive to n_0 when T is large. Chen *et al.* [2000, 2008] suggest that a good choice of n_0 is somewhere between 5 and 20.

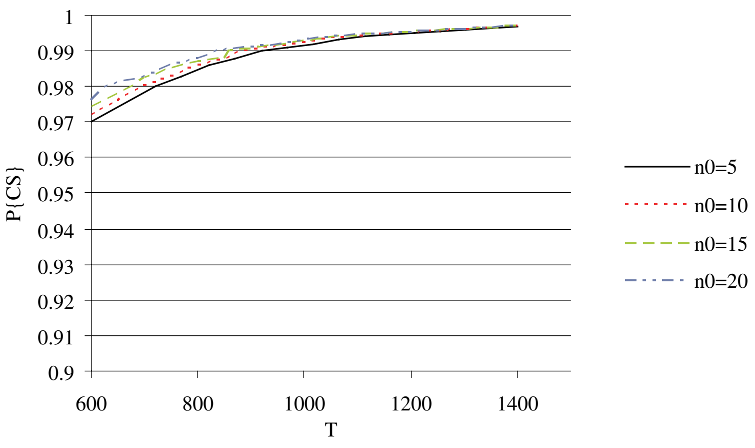


Figure 4.11. $P\{\text{CS}\}$ versus the computing budget T for different n_0 as OCBA is applied.

4.2.2. *One-time incremental computing budget, Δ*

The selection of Δ is typically problem-specific. Roughly speaking, Δ is more like the step size in typical optimization search algorithms. If the step size is too large, we may overshoot. On the other hand, it may take more steps to converge to an optimal solution if the step size is small. When applying the OCBA procedure, there is a similar trade-off between the required computation cost for the OCBA part and the resulting performance, particularly at the initial stage of simulation when the estimated means and variances are still not very accurate. A large Δ may result in a poor estimate of the predictive *APCS*, and OCBA may not be able to generate a good budget allocation. Thus, we may waste some computational efforts if Δ is too large. On the other hand, if Δ is too small, we need to solve the budget allocation problem given in the ALLOCATION step frequently, incurring higher computation cost for the OCBA part.

We test the OCBA approach using the worker allocation example for different values of Δ. To have a fair comparison, in Table 4.9, we report both the achieved $P\{\text{CS}\}$ and the total number of times the OCBA procedure is executed. We see that when Δ increases to 100, the number of times the budget allocation problem is solved is much smaller. However, the resulting $P\{\text{CS}\}$ is slightly lower.

Table 4.9. The achieved $P\{\text{CS}\}$ and the total number of times required to call the OCBA procedure for $n_0 = 10$.

	$\Delta = 1$		$\Delta = 5$	
T	$P\{\text{CS}\}$	# of OCBA	$P\{\text{CS}\}$	# of OCBA
300	90.3%	200	90.0%	40
500	96.4%	400	96.2%	80
700	98.5%	600	98.2%	120
900	99.4%	800	99.2%	160
1100	99.8%	1,000	99.6%	200
1300	99.9%	1,200	99.8%	240
1500	99.9%	1,400	99.9%	280
	$\Delta = 20$		$\Delta = 100$	
300	89.8%	10	89.0%	2
500	96.0%	20	95.4%	4
700	97.9%	30	97.5%	6
900	99.0%	40	98.6%	8
1100	99.4%	50	99.1%	10
1300	99.5%	60	99.3%	12
1500	99.8%	70	99.4%	14

A suggested choice for Δ is a number smaller than 100 or 10% of the simulated designs, whichever is smaller. Note that the computation time for solving the optimization problem of budget allocation in the ALLOCATION step depends mainly on the number of designs, k, and is independent with the complexity of the simulated system. Many real-life problems are much more complicated than a two-node system in our example. As a result, the computation cost for executing the simulation is usually much higher than the computation time for solving OCBA. Therefore, it is advisable to select a smaller Δ when dealing with more complicated systems. In this case, Δ should be as small as possible and can even be set as 1.

4.2.3. *Rounding off N_i to integers*

The resulting N_i in the ALLOCATE step based on Chapter 3's Equation (3.20), is a continuous number that must be rounded to an integer. There are several possible ways to do so. Ideally we want the rounded integers as close to the continuous numbers as possible.

In the numerical experiments presented in Section 4.1, let N_i be rounded to the nearest integer such that the summation of additional simulation replications for all designs equals Δ. By ensuring the total simulation replications for each iteration remain the same, it is easier to manage computing budget and have a fair comparison with other allocation procedures.

In general, we have found numerically that the performance of OCBA procedures is not sensitive to how we round N_i, probably due to the robustness of a sequential procedure.

To round off N_i to nearest integers while ensuring the summation of additional simulation replications for all designs equals Δ, the following algorithm can be applied. Let I_i be the integer part of N_i and D_i be the decimal point of N_i.

Step 1. $I_i = \lceil N_i \rceil$ and $D_i = N_i - I_i$, for all i.
Step 2. Calculate what the remaining computing budget (r) is if we take only the integer part, i.e., $r = \Delta - \sum_i I_i$.
Step 3. Find the set $\mathrm{S}(r) \equiv \{i : D_i$ is among the r highest values$\}$. Increase the computing budget by 1 for design $i \in \mathrm{S}(m)$, i.e., $N_i = I_i + 1$, if $i \in \mathrm{S}(m)$; $N_i = I_i$, otherwise.

Intuitively, this algorithm takes the integer part of each N_i, and then allocates the remaining budget to those designs which have higher decimal parts.

4.2.4. *Variance*

The allocation given in OCBA theorems assumes known variances. The OCBA sequential algorithm estimates these quantities using the updated sample variances. As more simulation replications are iteratively allocated to each design, the variance estimation improves. As discussed earlier, to avoid poor estimation at the beginning, n_0 should not be too small (we suggest $n_0 \geq 5$). Also, it is wise to avoid large Δ to prevent a poor allocation before a correction can be made in the next iteration, which is particularly important in the early stages. Our numerical testing indicates that the performance of the OCBA procedure is not sensitive to the choice of n_0 and Δ if the general guidelines are followed, and the impact of approximating variance by sample variance is not significant.

There is an alternative formula for the calculation of sample standard deviation in the UPDATE step:

$$s_i = \sqrt{\frac{1}{N_i^l - 1}\left(\sum_{j=1}^{N_i^l} X_{ij}^2 - N_i^l \bar{J}_i\right)}.$$

There is a small advantage to this alternative. It is not necessary to store the value of X_{ij} for all replications. To calculate $\sum X_{ij}^2$ and $\bar{J}_i$, we only need to accumulate X_{ij}^2 and X_{ij} and can throw away X_{ij} for each i and j after the accumulation step.

4.2.5. *Finite computing budget and normality assumption*

Although the OCBA allocation schemes are developed by taking $T \to \infty$, the numerical results presented in this chapter indicate that the corresponding allocation in the OCBA procedure works very efficiently for small T, as well. In addition, even though the simulation output is assumed to be normally distributed in deriving the allocation rule, the numerical testing indicates that the OCBA procedure works equally effectively by when the normality assumption is not valid.

Chapter 5

Selecting An Optimal Subset

Instead of selecting the best design as in previous chapters, we consider a class of subset selection problems in simulation optimization or ranking and selection. The objective is to identify the top-m out of k designs based on simulated output. In some cases, it is more useful to provide a set of good designs than a single best design for the decision-maker to choose, because he/she may have other concerns which are not modeled in the simulation. Such efficient subset selection procedures are also beneficial to some recent developments in simulation optimization that require the selection of an "elite" subset of good candidate solutions in each iteration of the algorithm, such as an evolutionary population-based algorithm. A subset with good performing solutions will result in an update that leads the search in a promising direction.

Most traditional subset selection procedures are conservative and inefficient. Using the optimal computing budget allocation framework, we formulate the problem as that of maximizing the probability of correctly selecting all of the top-m designs subject to a constraint on the total number of simulation replications available. For an approximation of this correct selection probability, we present an asymptotically optimal allocation and an easy-to-implement sequential allocation procedure. Numerical experiments indicate that the resulting allocations are superior to other methods in the literature that we tested, and the relative efficiency increases for larger problems. Some of the details can also be found in Chen *et al.* [2008].

5.1. Introduction and Problem Statement

Most of recent research development in multiple comparison or ranking and selection has focused on identifying the best design. Some traditional "subset selection" procedures aim at identifying a subset that *contains* the best design, dating back to Gupta [1965], who presented a single-stage procedure for producing a subset (of random size) containing the best design with a specified probability. Extensions of this work relevant to the simulation setting include Sullivan and Wilson [1989], who derive a two-stage subset selection procedure that determines a subset of maximum size m that, with a specified probability, contains at least one design whose mean response is within a pre-specified distance from the optimal mean response. This indifference zone procedure approach results in a subset of random size. The primary motivation for such procedures is *screening*, whereby the selected subset can be scrutinized further to find the single optimum.

Instead of selecting the very best design from a given set or finding a subset that is highly likely to contain the best design, our objective is to find *all* top-m designs, where $m > 1$. Koenig and Law [1985], who along the lines of the procedure in Dudewicz and Dalal [1975], developed a two-stage procedure for selecting all the m best designs (see also Section 10.4 of Law & Kelton 2000 for an extensive presentation of the problem and procedure). However, the number of additional simulation replications for the second stage is computed based on a least favorable configuration, resulting in very conservative allocations, so that the required computational cost is much higher than actually needed. While several procedures have been developed to enhance the efficiency of ranking and selection, most of them focus on selecting the single best, as presented in previous chapters.

Unlike traditional frequentist approaches constrained with least favorable configuration, Chen *et al.* [2008] developed an effective OCBA approach for selecting all the m best designs. As motivated in

Chapter 2, to ensure a high probability of correct selection, a larger portion of the computing budget should be allocated to those designs that are critical in the process of identifying the top-m designs. Ideally we want to allocate the computing budget in a most efficient way. Conservative least favorable configuration is no longer required, so the efficiency can be dramatically enhanced.

In addition to the notation given in previous chapters, we introduce the following notation:

$m =$	number of top designs to be selected in the optimal subset,
$S_m =$	set of m (distinct) indices indicating designs in selected subset.

The objective is to find a simulation budget allocation that maximizes the probability of selecting the *optimal subset*, defined as the set of m ($<k$) best designs, for m a fixed number. Without loss of generality, we will take as the m best designs those designs with the m smallest means. Note that rank order within the subset is not part of the objective.

In the selection, we will take S_m to be the m designs with the smallest *sample* means. Let $\bar{J}_{i_r}$ be the r-th smallest (order statistic) of $\{\bar{J}_1, \bar{J}_2, \ldots, \bar{J}_k\}$, i.e., $\bar{J}_{i_1} \leq \bar{J}_{i_2} \leq \cdots \leq \bar{J}_{i_k}$. Then, the selected subset is given by

$$S_m \equiv \{i_1, i_2, \ldots, i_m\}. \tag{5.1}$$

Since the simulation output is stochastic, the set S_m is a random set. In terms of our notation, the correct selection event is defined by S_m containing all of the m smallest mean designs:

$$\mathrm{CS}_m \equiv \left\{ \bigcap_{i \in S_m} \bigcap_{j \notin S_m} (J_i \leq J_j) \right\} = \left\{ \max_{i \in S_m} J_i \leq \min_{i \notin S_m} J_i \right\}. \tag{5.2}$$

The optimal computing budget allocation (OCBA-m) problem is given by

$$\max_{N_1,\ldots,N_k} \; P\{\mathrm{CS}_m\}$$
$$\text{s.t. } N_1 + N_2 + \cdots + N_k = T. \tag{5.3}$$

Here $N_1 + N_2 + \cdots + N_k$ denotes the total computational cost, assuming the simulation execution times for different designs are roughly the same. This formulation implicitly assumes that the computational cost of each replication is constant across designs. Please note that this assumption can be relaxed easily without changing the solution. See Section 3.5 for details.

The optimal computing budget allocation problems given in Chapter 3 is actually a special case of (5.3) with $m = 1$. For notational simplification, we will drop the "m" in $P\{\mathrm{CS}_m\}$ in the remaining chapter.

5.2. Approximate Asymptotically Optimal Allocation Scheme

Our approach is developed based on the same Bayesian setting presented in Chapter 3. If one prefers classical statistical model, a similar optimal allocation can also be derived using the same idea. An analogy is presented in Section 3.7 for selecting the single best.

In the Bayesian framework, the mean of the simulation output for each design, J_i, is assumed unknown and treated as a random variable. After the simulation is performed, a posterior distribution for the unknown mean J_i, $p(J_i \,|\, L(\theta_i, \omega_{ij}),\ j = 1, 2, \ldots, N_i)$, is constructed based on two pieces of information: (i) prior knowledge of the system's performance, and (ii) current simulation output. Thus, the probability of correct selection defined by Equation (5.2) is given by

$$P\{\mathrm{CS}\} = P\{\tilde{J}_i \le \tilde{J}_j, \text{ for all } i \in S_m \text{ and } j \notin S_m\}, \tag{5.4}$$

where $\tilde{J}_i$, $i = 1, \ldots, k$, denotes the random variable whose probability distribution is the posterior distribution of design i. As in Chapter 3, we assume that the unknown mean J_i has a conjugate normal prior distribution and consider non-informative prior distributions, which implies that no prior knowledge is available about the performance of any design before conducting the simulations, in which case the posterior distribution of J_i is

$$\tilde{J}_i \sim N\left(\bar{J}_i, \frac{\sigma_i^2}{N_i}\right). \tag{5.5}$$

After the simulation is performed, $\bar{J}_i$ can be calculated, σ_i^2 can be approximated by the sample variance, and the $P\{\text{CS}\}$ given by Equation (5.4) can then be estimated using Monte Carlo simulation. However, since estimating $P\{\text{CS}\}$ via Monte Carlo simulation is time-consuming and the purpose of budget allocation is to improve simulation efficiency, we adopt an approximation of $P\{\text{CS}\}$ using a lower bound.

For a constant c,

$$\begin{aligned} P\{\text{CS}\} &= P\{\tilde{J}_i \le \tilde{J}_j, \text{ for all } i \in S_m \text{ and } j \notin S_m\} \\ &\ge P\{\tilde{J}_i \le c \text{ and } \tilde{J}_j \ge c, \text{ for all } i \in S_m \text{ and } j \notin S_m\} \\ &= \prod_{i \in S_m} P\{\tilde{J}_i \le c\} \prod_{i \notin S_m} P\{\tilde{J}_i \ge c\} \equiv APCSm, \end{aligned} \tag{5.6}$$

where the last line is due to independence across designs. We refer to this lower bound for $P\{\text{CS}\}$, which can be computed easily and eliminates the need for extra Monte Carlo simulation, as the *Approximate Probability of Correct Selection for m best* (*APCSm*). The value for c will be between $\bar{J}_{i_m}$ and $\bar{J}_{i_{m+1}}$, and the rationale for this will be explained in Section 5.2.1. Using the approximation given by Equation (5.6), the OCBA-m problem (5.3) becomes

$$\begin{aligned} &\max_{N_1,\ldots,N_k} \prod_{i \in S_m} P\{\tilde{J}_i \le c\} \prod_{i \notin S_m} P\{\tilde{J}_i \ge c\} \\ &\text{s.t. } N_1 + N_2 + \cdots + N_k = T. \end{aligned} \tag{5.7}$$

Now we solve OCBA problem (5.7), assuming the variables $\{N_i\}$ are continuous. For notation simplification, we define the variable $\delta_i = \bar{J}_i - c$, $i = 1, 2, \ldots, k$.

For $i \in S_m$,

$$\begin{aligned} P(\tilde{J}_i \le c) &= \int_{-\infty}^{0} \frac{1}{\sqrt{2\pi}(\sigma_i/\sqrt{N_i})} e^{\frac{-(x-\delta_i)^2}{2(\sigma_i^2/N_i)}} dx \\ &= \int_{\frac{\delta_i}{(\sigma_i/\sqrt{N_i})}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} dt = \Phi\left(\frac{-\delta_i}{\sigma_i/\sqrt{N_i}}\right), \end{aligned}$$

and for $i \notin S_m$,

$$P(\tilde{J}_i \geq c) = \int_0^{\infty} \frac{1}{\sqrt{2\pi}(\sigma_i/\sqrt{N_i})} e^{\frac{-(x-\delta_i)^2}{2(\sigma_i^2/N_i)}} dx = \Phi\left(\frac{\delta_i}{\sigma_i/\sqrt{N_i}}\right),$$

where $\Phi(x)$ is the standard normal cumulative distribution function. Now let F be the Lagrangian relaxation of (5.7), with Lagrange multiplier λ:

$$\begin{aligned} F &= \prod_{i \in S_m} P\{\tilde{J}_i \leq c\} \cdot \prod_{i \notin S_m} P\{\tilde{J}_i \geq c\} - \lambda \left(\sum_{i=1}^{k} N_i - T\right) \\ &= \prod_{i \in S_m} \Phi\left(\frac{-\delta_i}{\sigma_i/\sqrt{N_i}}\right) \cdot \prod_{i \notin S_m} \Phi\left(\frac{\delta_i}{\sigma_i/\sqrt{N_i}}\right) - \lambda\left(\sum_{i=1}^{k} N_i - T\right). \end{aligned} \tag{5.8}$$

To find the stationary point of the Lagrangian function, we set $\frac{\partial F}{\partial N_i} = 0$ for each value of i.

For $i \in S_m$,

$$\begin{aligned} \frac{\partial F}{\partial N_i} = \prod_{\substack{j \in S_m \\ j \neq i}} P\{\tilde{J}_j \leq c\} \cdot \prod_{j \notin S_m} P\{\tilde{J}_j \geq c\} \\ \cdot \frac{-1}{2}\varphi\left(\frac{\delta_i}{\sigma_i/\sqrt{N_i}}\right) \frac{\delta_i}{\sigma_i} N_i^{-1/2} - \lambda = 0, \end{aligned} \tag{5.9}$$

where $\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}$ is the standard normal probability density function.

For $i \notin S_m$,

$$\begin{aligned} \frac{\partial F}{\partial N_i} = \prod_{j \in S_m} P\{\tilde{J}_j \leq c\} \cdot \prod_{\substack{j \notin S_m \\ j \neq i}} P\{\tilde{J}_j \geq c\} \\ \cdot \frac{1}{2}\varphi\left(\frac{\delta_i}{\sigma_i/\sqrt{N_i}}\right) \frac{\delta_i}{\sigma_i} N_i^{-1/2} - \lambda = 0. \end{aligned} \tag{5.10}$$

Also, $\frac{\partial F}{\partial \lambda} = 0$ returns the budget constraint $\sum_{i=1}^{k} N_i - T = 0$.

To examine the relationship between N_i and N_j for $i \neq j$, we consider three cases:

(1) $i \in S_m$, and $j \notin S_m$:

Equating the expressions in Equations (5.9) and (5.10),

$$\prod_{\substack{r\in S_m\\ r\neq i}} P\{\tilde{J}_r \leq c\} \cdot \prod_{r\notin S_m} P\{\tilde{J}_r \geq c\} \cdot \frac{-1}{2}\varphi\left(\frac{\delta_i}{\sigma_i/\sqrt{N_i}}\right)\frac{\delta_i}{\sigma_i}N_i^{-1/2} - \lambda$$

$$= \prod_{r\in S_m} P\{\tilde{J}_r \leq c\} \cdot \prod_{\substack{r\notin S_m\\ r\neq j}} P\{\tilde{J}_r \geq c\}$$

$$\cdot \frac{1}{2}\varphi\left(\frac{\delta_j}{\sigma_j/\sqrt{N_j}}\right)\frac{\delta_j}{\sigma_j}N_j^{-1/2} - \lambda.$$

Simplifying,

$$P\{\tilde{J}_j \geq c\} \cdot e^{\frac{-\delta_i^2}{2(\sigma_i^2/N_i)}}\frac{-\delta_i}{\sigma_i}N_i^{-1/2} = P\{\tilde{J}_i \leq c\} \cdot e^{\frac{-\delta_j^2}{2(\sigma_j^2/N_j)}}\frac{\delta_j}{\sigma_j}N_j^{-1/2}. \tag{5.11}$$

Taking the log on both sides,

$$\log(P\{\tilde{J}_j \geq c\}) - \frac{\delta_i^2 N_i}{2\sigma_i^2} + \log\left(\frac{-\delta_i}{\sigma_i}\right) - \frac{1}{2}\log(N_i)$$

$$= \log(P\{\tilde{J}_i \leq c\}) - \frac{\delta_j^2 N_j}{2\sigma_j^2} + \log\left(\frac{\delta_j}{\sigma_j}\right) - \frac{1}{2}\log(N_j).$$

Now, we consider the asymptotic limit $T \to \infty$ with $N_i = \alpha_i T$, $\sum_{i=1}^k \alpha_i = 1$. Substituting for N_i and dividing by T yields

$$\frac{1}{T}\log(P\{\tilde{J}_j \geq c\}) - \frac{\delta_i^2}{2\sigma_i^2}\alpha_i + \frac{1}{T}\log\left(\frac{-\delta_i}{\sigma_i}\right) - \frac{1}{2T}\log(\alpha_i T)$$

$$= \frac{1}{T}\log\left(P\{\tilde{J}_i \leq c\}\right) - \frac{\delta_j^2}{2\sigma_j^2}\alpha_j + \frac{1}{T}\log\left(\frac{\delta_j}{\sigma_j}\right) - \frac{1}{2T}\log(\alpha_j T),$$

and then taking $T \to \infty$ yields

$$\frac{\delta_i^2}{\sigma_i^2}\alpha_i = \frac{\delta_j^2}{\sigma_j^2}\alpha_j.$$

Therefore, we obtain the ratio between α_i and α_j or between N_i and N_j as:

$$\frac{N_i}{N_j} = \frac{\alpha_i}{\alpha_j} = \left(\frac{\sigma_i/\delta_i}{\sigma_j/\delta_j}\right)^2. \tag{5.12}$$

Since $\delta_i = \bar{J}_i - c$, $i = 1, 2, \ldots, k$, it is clear that the optimal allocation depends on the value of c (see Section 5.2.1 for determining a good value), and thus the $m = 1$ case does not in general reduce to the original OCBA allocation for selecting the best design.

(2) Both $i, j \in S_m$ and $i \neq j$:

From Equation (5.9),

$$\frac{\partial F}{\partial N_i} = \frac{\partial F}{\partial N_j} = 0$$

yields

$$\begin{aligned}
&\prod_{\substack{r\in S_m\\ r\neq i}} P\{\tilde{J}_r \le c\} \cdot \prod_{r\notin S_m} P\{\tilde{J}_r \ge c\} \\
&\qquad \cdot \frac{-1}{2}\varphi\left(\frac{\delta_i}{\sigma_i/\sqrt{N_i}}\right)\frac{\delta_i}{(\sigma_i^2/N_i)}\frac{\sigma_i}{N_i^{3/2}} - \lambda \\
&= \prod_{\substack{r\in S_m\\ r\neq j}} P\{\tilde{J}_r \le c\} \cdot \prod_{r\notin S_m} P\{\tilde{J}_r \ge c\} \\
&\qquad \cdot \frac{-1}{2}\varphi\left(\frac{\delta_j}{\sigma_j/\sqrt{N_j}}\right)\frac{\delta_j}{(\sigma_j^2/N_j)}\frac{\sigma_j}{N_j^{3/2}} - \lambda.
\end{aligned}$$

Then,

$$P\{\tilde{J}_j \le c\} \cdot e^{\frac{-\delta_i^2}{2(\sigma_i^2/N_i)}}\frac{-\delta_i}{\sigma_i}N_i^{-1/2} = P\{\tilde{J}_i \le c\} \cdot e^{\frac{-\delta_j^2}{2(\sigma_j^2/N_j)}}\frac{-\delta_j}{\sigma_j}N_j^{-1/2},$$

which is the same as Equation (5.11). Following the analogous derivation that led to Equation (5.12) yields the same result

$$\frac{N_i}{N_j} = \frac{\alpha_i}{\alpha_j} = \left(\frac{\sigma_i/\delta_i}{\sigma_j/\delta_j}\right)^2. \tag{5.13}$$

(3) $i, j \notin S_m$, and $i \neq j$:
From Equation (5.10),

$$\frac{\partial F}{\partial N_i} = \frac{\partial F}{\partial N_j} = 0$$

yields

$$P\{\tilde{J}_j \leq c\} \cdot e^{\frac{-\delta_i^2}{2(\sigma_i^2/N_i)}} \frac{-\delta_i}{\sigma_i} N_i^{-1/2} = P\{\tilde{J}_i \leq c\} \cdot e^{\frac{-\delta_j^2}{2(\sigma_j^2/N_j)}} \frac{-\delta_j}{\sigma_j} N_j^{-1/2}.$$

Again, following the same derivation procedures as in the previous two cases leads to the same result

$$\frac{N_i}{N_j} = \frac{\alpha_i}{\alpha_j} = \left(\frac{\sigma_i/\delta_i}{\sigma_j/\delta_j}\right)^2. \tag{5.14}$$

In all of the above three cases, we obtain the same locally optimal solution to the Lagrangian relaxation of the OCBA-m problem (5.7) and therefore have the following result.

Theorem 5.1. *Given a total number of simulation replications T to be allocated to k competing designs whose performance is depicted by random variables with means $J_1, J_2, \ldots, J_k$, and finite variances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_k^2$ respectively, as $T \to \infty$, the* Approximate Probability of Correct Selection for m best (*APCSm*) *can be asymptotically maximized when*

$$\frac{N_i}{N_j} = \left(\frac{\sigma_i/\delta_i}{\sigma_j/\delta_j}\right)^2, \tag{5.15}$$

for any $i, j \in \{1, 2, \ldots, k\}$, and $i \neq j$, where $\delta_i = \bar{J}_i - c$, for c a constant.

Remark 5.1 (Signal to Noise Ratio). Note that Equation (5.15) can be rewritten as

$$\frac{N_i}{N_j} = \left(\frac{\delta_j/\sigma_j}{\delta_i/\sigma_i}\right)^2, \tag{5.16}$$

which is very similar to the signal to noise ratio equation given in Remark 3.3 that is the result of selecting the single best.

Similar with Remark 3.3, $\frac{\delta_i}{\sigma_i}$ can be considered as a signal to noise ratio for design i. In the problem of selecting the top-m, the higher $\frac{\delta_i}{\sigma_i}$, the more confident we are about whether design i is in the optimal subset or not. Intuitively, we should spend less computing budget on those designs for which we are highly confident, i.e., high signal to noise ratio. The allocated computing budget is inversely proportional to the square of the signal to noise ratio.

It is interesting to see that the signal to noise relation for selecting the top-m is the same as that for selecting the single best, except that the definitions of δ are different.

5.2.1. *Determination of c value*

The parameter c impacts the quality of the approximation *APCSm* to $P\{\text{CS}\}$. Since *APCSm* is a lower bound of $P\{\text{CS}\}$, choosing c to make *APCSm* as large as possible is likely to provide a better approximation of *APCSm* to $P\{\text{CS}\}$. However, determining c to maximize *APCSm* can be time-consuming. We offer a simple heuristic as an approximation in this book. This certainly can be further improved.

Figure 5.1 is provided to help explain our choice of c, by giving an example of probability density functions for $\tilde{J}_i$, $i = 1, 2, \ldots, k$. Note that *APCSm* is a product of $P\{\tilde{J}_i \leq c\}$ for $i \in S_m$ and $P\{\tilde{J}_i \geq c\}$ for $i \notin S_m$. Consider the equal variance case $\text{Var}(\tilde{J}_{i_1}) = \text{Var}(\tilde{J}_{i_2}) = \cdots =$

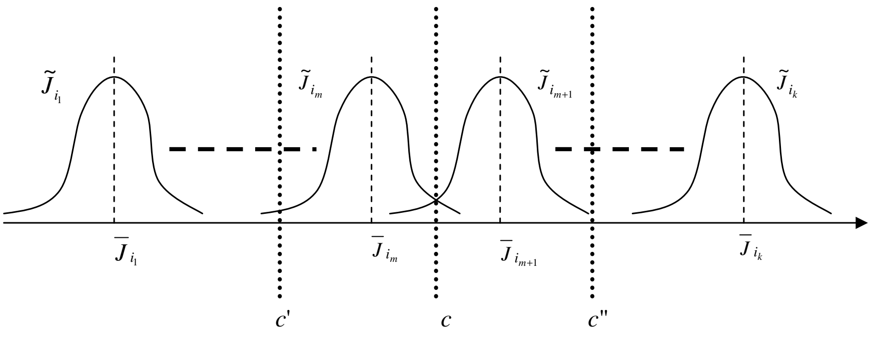


Figure 5.1. An example of probability density functions for $\tilde{J}_i$, $i = 1, 2, \ldots, k$, $c' < \bar{J}_{i_m} < c < \bar{J}_{i_{m+1}} < c''$.

$\mathrm{Var}(\tilde{J}_{i_k})$, where for any value of c, $P\{\tilde{J}_{i_1} \le c\} > P\{\tilde{J}_{i_2} \le c\} > \cdots > P\{\tilde{J}_{i_m} \le c\}$, and $P\{\tilde{J}_{i_{m+1}} \ge c\} < P\{\tilde{J}_{i_{m+2}} \ge c\} < \cdots < P\{\tilde{J}_{i_k} \ge c\}$. To prevent *APCSm* from being small, we want to choose c to avoid any of the product terms being too small, especially $P\{\tilde{J}_{i_m} \le c\}$ and $P\{\tilde{J}_{i_{m+1}} \ge c\}$, because one of these two terms is usually the smallest in the product, depending on the value of c. Thus, a good choice of c lies between $\bar{J}_{i_m}$ and $\bar{J}_{i_{m+1}}$, because

(i) if $c = c' < \bar{J}_{i_m}$, then $P\{\tilde{J}_{i_m} < c'\} < 0.5$, and this term decreases with decreasing c', resulting in a negative impact on *APCSm*;
(ii) similarly, if $c = c'' > \bar{J}_{i_{m+1}}$, then $P\{\tilde{J}_{i_{m+1}} > c''\} < 0.5$, and this term decreases with increasing c''.

With these considerations, one would like to maximize both $P\{\tilde{J}_{i_m} \le c\}$ and $P\{\tilde{J}_{i_{m+1}} \ge c\}$. Chen *et al.* (2008) chose to maximize the product of $P\{\tilde{J}_{i_m} \le c\}$ and $P\{\tilde{J}_{i_{m+1}} \ge c\}$. Define $\hat{\sigma}_i \equiv \sigma_i/\sqrt{N_i}$. Then

$$P\{\tilde{J}_{i_m} \le c\}P\{\tilde{J}_{i_{m+1}} \ge c\} = \Phi\left(\frac{c - \bar{J}_{i_m}}{\hat{\sigma}_{i_m}}\right)\Phi\left(\frac{\bar{J}_{i_{m+1}} - c}{\hat{\sigma}_{i_{m+1}}}\right). \tag{5.17}$$

Following the same approach as used to establish Theorem 5.1, this quantity is asymptotically maximized when

$$c = \frac{\hat{\sigma}_{i_{m+1}}\bar{J}_{i_m} + \hat{\sigma}_{i_m}\bar{J}_{i_{m+1}}}{\hat{\sigma}_{i_m} + \hat{\sigma}_{i_{m+1}}}. \tag{5.18}$$

We use this value of c in our implementation, and the numerical experiments show that it results in good performance.

Remark 5.2. Note that c falls between designs i_m and i_{m+1} and can be considered as the border which divides the best top-m subset from the remaining designs. Namely, c is the border separating good designs and non-good designs. Therefore, $|\delta_i|$ is equivalent to the distance between design i and the border. From the computing budget allocation perspective, to identify which designs belong to top-m, the "critical" designs are those which are near the border. The further away from the border, the more we are confident that they are either very good or very bad designs. We should spend less computing budget on those with large $|\delta_i|$. On the other hand, we should spend more efforts on those designs which are near the border. This is consistent with the discussion in Remark 5.1. The allocated computing budget is inversely proportional to the square of "δ" or, more specifically, the signal to noise ratio.

5.2.2. *Sequential allocation scheme*

We present an effective sequential allocation procedure by utilizing Theorem 5.1. The objective is to identify the top-m design using a minimum simulation budget or to maximize the probability of selecting all the top-m designs using a same computing budget (T). The allocation given by Theorem 5.1 assumes known variances and independence of estimated sample means across designs. In practice, a sequential algorithm is used to estimate these quantities using the updated sample variances. Furthermore, the "constant" c and sample means are also updated during each iteration. Each design is initially simulated with n_0 replications in the first stage, and additional replications are allocated incrementally with Δ replications to be allocated in each iteration, in which we want to maximize the effectiveness of determining the best top-m. This procedure is continued until the total budget T is exhausted or the estimated $APCSm$ is sufficiently high.

OCBA-m procedure

```
INPUT          $k$, $m$, $T$, $\Delta$, $n_0 (T - kn_0$ a multiple of $\Delta$ and
               $n_0 \geq 5)$;
INITIALIZE     $l \leftarrow 0$;
               Perform $n_0$ simulation replications for all
               designs; $N_1^l = N_2^l = \cdots = N_k^l = n_0$.
LOOP           WHILE $\sum_{i=1}^{k} N_i^l < T$ DO
    UPDATE     Calculate sample means $\bar{J}_i = \frac{1}{N_i^l} \sum_{j=1}^{N_i^l}$
               $L(\theta_i, \omega_{ij})$, and sample standard deviation $s_i =$
               $\sqrt{\frac{1}{N_i^l - 1} \sum_{j=1}^{N_i^l} (L(\theta_i, \omega_{ij}) - \bar{J}_i)^2}$, $i = 1, \ldots, k$,
               using the new simulation output; determine
```

the subset S_m using the ranking of $\bar{J}_i$; compute $\hat{\sigma}_i = \frac{s_i}{\sqrt{N_i^l}}$, $i = 1, \ldots, k$, and $c = \frac{\hat{\sigma}_{i_{m+1}} \bar{J}_{i_m} + \hat{\sigma}_{i_m} \bar{J}_{i_{m+1}}}{\hat{\sigma}_{i_m} + \hat{\sigma}_{i_{m+1}}}$; update $\delta_i = \bar{J}_i - c$, $i = 1, \ldots k$.

ALLOCATE Increase the computing budget by Δ and calculate the new budget allocation, N_1^{l+1}, $N_2^{l+1}, \ldots, N_k^{l+1}$, according to

$$\frac{N_1^{l+1}}{\left(\frac{S_1}{\delta_1}\right)^2} = \frac{N_2^{l+1}}{\left(\frac{S_2}{\delta_2}\right)^2} = \cdots = \frac{N_k^{l+1}}{\left(\frac{S_k}{\delta_k}\right)^2}.$$

SIMULATE Perform additional $\max(N_i^{l+1} - N_i^l, 0)$ simulations for design i, $i = 1, \ldots, k$; $l \leftarrow l + 1$.

END OF LOOP

In the above procedure, the stopping criterion is that the total budget T is exhausted. Alternatively, one may want to minimize overall computing cost (total number of simulation replications) in order to achieve a desired $P\{\text{CS}\}$, such as 95%. In the later case, the "Loop While $\sum_{i=1}^{k} N_i^l < T$ Do" step is replaced with "Loop While *APCSm* $< P^*$ Do", where P^* is the desired correction probability. Based on the Bayesian framework in Section 3.1 and Equations (3.2) and (3.3),

$$\begin{aligned} APCSm &= \prod_{i \in S_m} P\{\tilde{J}_i \le c\} \prod_{i \notin S_m} P\{\tilde{J}_i \ge c\} \\ &= \prod_{i \in S_m} \Phi\left(\frac{c - \bar{J}_i}{\sqrt{\frac{s_i^2}{N_i^l}}}\right) * \prod_{i \notin S_m} \Phi\left(\frac{\bar{J}_i - c}{\sqrt{\frac{s_i^2}{N_i^l}}}\right). \end{aligned} \quad (5.19)$$

For a good choice of the parameters in the OCBA-m procedure such as Δ and n_0, please refer to Chapter 4.

Note that this OCBA-m procedure is designed to select all of the top-m designs when $m \ge 2$. For the $m = 1$ case, the OCBA allocation given in Chapter 3 is different from the OCBA-m due to different approximations made. Our empirical studies show that both the original OCBA procedure and OCBA-m procedure work very well when $m = 1$, though the original OCBA procedure performs slightly better.

Alternative OCBA-m procedure

As discussed in Section 4.1.1, when the simulation is relatively expensive, the computation cost of the ALLOCATION step becomes negligible. In this case, Δ should be as small as possible and can be set as 1. To avoid the numerical error occurred when rounding N_i^{l+1} into an integer, we suggest a numerically robust alternative. The ALLOCATE and SIMULATE steps are revised as follows.

ALLOCATE Increase the computing budget by $\Delta = 1$ and calculate the new budget allocation, N_1^{l+1}, $N_2^{l+1}, \ldots, N_k^{l+1}$, according to

$$\frac{N_1^{l+1}}{\left(\frac{S_1}{\delta_1}\right)^2} = \frac{N_2^{l+1}}{\left(\frac{S_2}{\delta_2}\right)^2} = \cdots = \frac{N_k^{l+1}}{\left(\frac{S_k}{\delta_k}\right)^2};$$

leave N_i^{l+1} as a decimal number and find $i^* = \arg\max_i (N_i^{l+1} - N_i^l)$.

SIMULATE Perform one additional simulation for design i^*; $N_{i*}^{l+1} = N_{i*}^l + 1$; $N_i^{l+1} = N_i^l$, for $i \neq i^*$; $l \leftarrow l + 1$.

The revision determines which design is most in need of an additional simulation, and then simulates that design for one additional replication. This procedure is iteratively continued until the total budget T is exhausted or the estimated $APCSm$ is sufficiently high.

5.3. Numerical Experiments

To compare the performance of the procedures, we carry out numerical experiments for several typical selection problems. Most of the numerical setting is the same as those presented in Chapter 4. Details of the numerical experiments can be found in Chen *et al.* (2008).

For notational simplicity, in this section, we assume $J_{[1]} < J_{[2]} < \cdots < J_{[k]}$, so design [1] is the best, and the optimal subset containing the top-m is $\{[1], [2], \ldots, [m]\}$ (but this is unknown *a priori*). It is a correct selection if the selected subset $S_m = \{[1], [2], \ldots, [m]\}$.

5.3.1. *Computing budget allocation procedures*

We test the OCBA-m algorithm by comparing it on several numerical experiments with different allocation procedures: Equal Allocation, which simulates all design alternatives equally; the Koenig and Law (1985) procedure denoted by KL; Proportional To Variance (PTV), which is a modification of KL that allocates replications proportional to the estimated variances; and the original OCBA allocation algorithm for selecting only the best design. For details of Equal Allocation, PTV, and OCBA procedures, please refer to Chapters 3 and 4. We briefly describe the KL procedure here.

The two-stage procedure of Koenig and Law (1985) selects a subset of specified size m, with probability at least P^*, so that the selected subset is exactly the actual subset with the best (smallest) expected values, provided that $J_{[m+1]} - J_{[m]}$ is no less than an indifference zone, d. As in our setting, the ordering within the selected subset does not matter.

In the first stage, all designs are simulated for n_0 replications. Based on the sample variance estimate (s_i^2) obtained from the first stage and given the desired correct selection probability P^*, the number of additional simulation replications for each design in the second stage is determined by:

$$N_i = \max(n_0 + 1, \lceil h_3^2 s_i^2 / d^2 \rceil), \quad \text{for } i = 1, 2, \ldots, k, \qquad (5.20)$$

where $\lceil \bullet \rceil$ is the integer "round-up" function, and h_3 is a constant that depends on k, m, P^*, and n_0.

5.3.2. *Numerical results*

In comparing the procedures, the measurement of effectiveness used is the $P\{\text{CS}\}$ estimated by the fraction of times the procedure successfully finds *all* the true m-best designs out of 100,000 independent experiments. Because this penalizes incorrect selections equally — e.g., a subset containing the top-1, top-2, ..., and top-$(m-1)$ designs and missing only the top-m design is treated no differently than a subset containing not a single one of the top-m designs — in our numerical experiments, we also include a second measure of selection quality, the so-called expected opportunity cost E[OC], where

$$\text{OC} \equiv \sum_{j=1}^{m} (J_{i_j} - J_{[j]}). \tag{5.21}$$

This measure penalizes particularly bad choices more than mildly bad choices. For example, when $m = 3$, a selection of {top-1, top-2, top-4} is better than {top-1, top-2, top-5}, and both are better than {top-1, top-3, top-5}. Note that OC returns a minimum value of 0 when all the top-m designs are correctly selected. The estimated E[OC] is the average of the OC estimates over the 100,000 independent experiments.

Each of the procedures simulates each of the k designs for $n_0 = 20$ replications initially (following recommendations in Koenig and Law 1985 and Law and Kelton 2000). KL allocates additional replications in a second stage (so the total number is not fixed *a priori*), whereas the other procedures allocate replications incrementally by $\Delta = 50$ each time until the total budget, T, is consumed. For each level of computing budget, we estimate the achieved $P\{\text{CS}\}$ and E[OC].

Since KL is a two-stage indifference-zone procedure, we must specify the values for the desired probability of correct selection, P^*, and the indifference zone d to satisfy the condition that $J_{[m+1]} - J_{[m]} \geq d$, where a smaller d implies a higher required computation cost based on Equation (5.21). In practice, the value of $J_{[m+1]}$ or $J_{[m]}$ is unknown beforehand, but for benchmarking purposes, we set $d = J_{[m+1]} - J_{[m]}$, which leads to the *minimum* computational requirement (or maximum efficiency) for the procedure. Since the required computation cost varies from one experiment to another, we will indicate the average number of total replications based on the 100,000 independent experiments. As is done for the other procedures, the resulting $P\{\text{CS}\}$ and E[OC] can also be estimated over the 100,000 independent experiments.

Experiment 5.1. Worker allocation problem

This numerical test is based on the worker allocation problem presented in Example 1.1 (also Experiment 4.1). In summary, this problem represents the challenge of allocating 31 workers within a two-stage queue where each stage of the queue can contain no less than 11 workers as shown in Figure 5.2.

Denote C_1 and C_2 as the numbers of workers allocated to nodes 1 and 2. Thus $C_1 + C_2 = 31$, $C_1 \geq 11$, and $C_2 \geq 11$. There are 10 alternative combinations of (C_1, C_2). We want to choose the best alternative of (C_1, C_2) so that the average system time for the first 100 customers is minimized. Since there is no closed-form analytical solution for the estimation of the system time, stochastic (discrete-event) simulation can be performed to derive a solution.

To characterize the performance of different procedures as a function of T, we vary T between 200 and 3000 for all of the procedures other than KL, and the estimated achieved $P\{\text{CS}\}$ and E[OC] as a function of T is shown in Figure 5.3. For KL, we test two cases $P^* = 0.9$ and $P^* = 0.95$. Since the corresponding estimated $P\{\text{CS}\}$ and E[OC] versus the average total simulation replications are well beyond the range of our figures, they are shown in Table 5.1.

We see that all procedures obtain a higher $P\{\text{CS}\}$ and lower E[OC] as the available computing budget increases. However, OCBA-m achieves the highest $P\{\text{CS}\}$ and lowest E[OC] for the same amount of computing budget. It is interesting to observe that OCBA, which performs significantly better than Equal Allocation and PTV

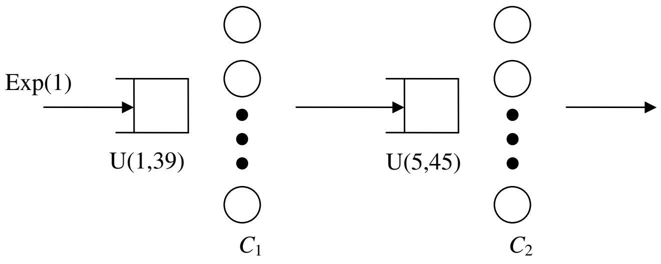


Figure 5.2. A two-stage queuing system where both C_1, and C_2 must be greater than 10.

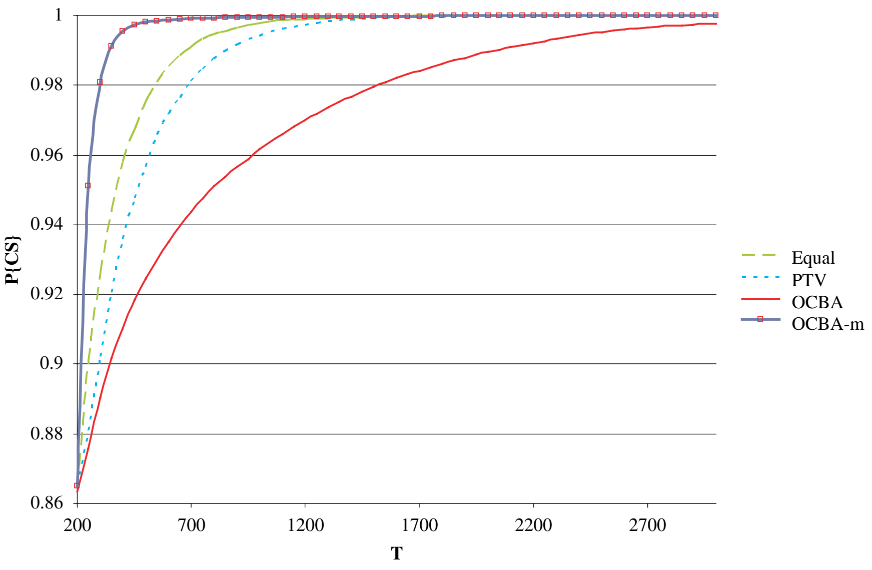


Figure 5.3(a). $P\{\text{CS}\}$ versus T using four sequential allocation procedures and KL for Experiment 5.1.

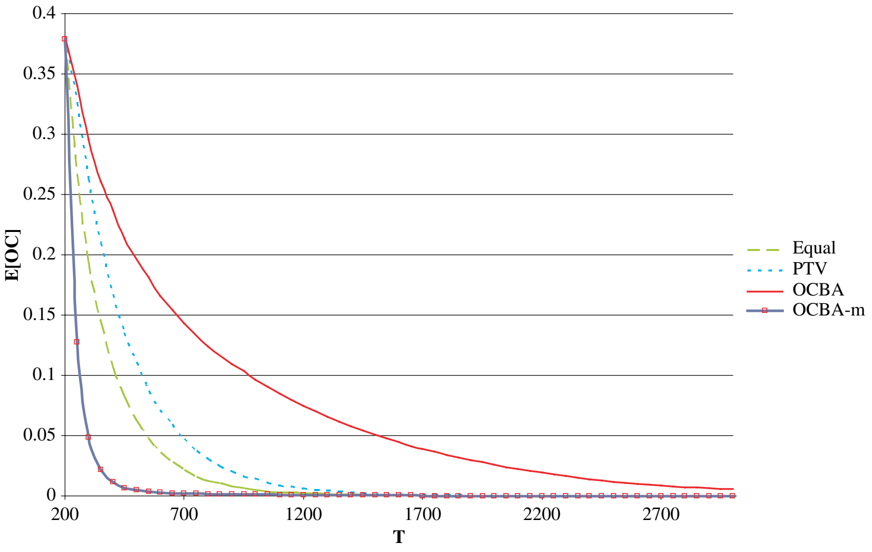


Figure 5.3(b). E[OC] versus T using four sequential allocation procedures and KL for Experiment 5.1.

Table 5.1. Results for KL procedure.

P^*	Avg. total # of replications	$P\{\text{CS}\}$	E[OC]
90%	5388	100%	0.0
95%	7007	100%	0.0

when the objective is to find the single best design, fares worse in this experiment than these two allocations when the objective is changed to finding all the top-3 designs. Specifically, the computation costs to attain $P\{\text{CS}\} = 0.95$ for OCBA-m, OCBA, Equal, and PTV are 350, 2050, 700, 900, respectively. KL achieves a substantially higher $P\{\text{CS}\}$ than the desired level (e.g., 1.0 for the target minimum of both $P^* = 0.9$ and $P^* = 0.95$) by spending a much higher computing budget than actually needed, consistent with the fact that typical two-stage indifference-zone procedures are conservative.

Experiment 5.2. (s, S) Inventory control problem

The second experiment is an (s, S) inventory policy problem which is given as Example 1.2 in Chapter 1. In summary, the system involves a single item under periodic review, full backlogging, and random lead times, with costs for ordering, on-hand inventory, and backlogging. The times between demands and the sizes of demands are random. The (s, S) policy specifies that if the on-hand inventory at the review point is at or below the level s, then an order is placed of an amount that would bring the inventory up to level S. The 10 inventory policies are defined by the parameters $(s_1, s_2, \ldots, s_{10}) = (20, 20, 20, 40, 40, 40, 60, 60, 60, 80)$ and $(S_1, S_2, \ldots, S_{10}) = (30, 40, 50, 50, 60, 70, 70, 80, 90, 90)$, respectively. The objective is to find the top-3 ($m = 3$) policies with minimum expected average inventory cost over 120 periods.

The test results shown in Figure 5.4 are similar to those in previous experiment, in that OCBA-m is clearly the top performer again; however, this time OCBA is the runner up by a slight margin. The computation costs to attain $P\{\text{CS}\} = 0.95$ for OCBA-m, OCBA, Equal, and PTV are 500, 1200, 1650, 1350, respectively.

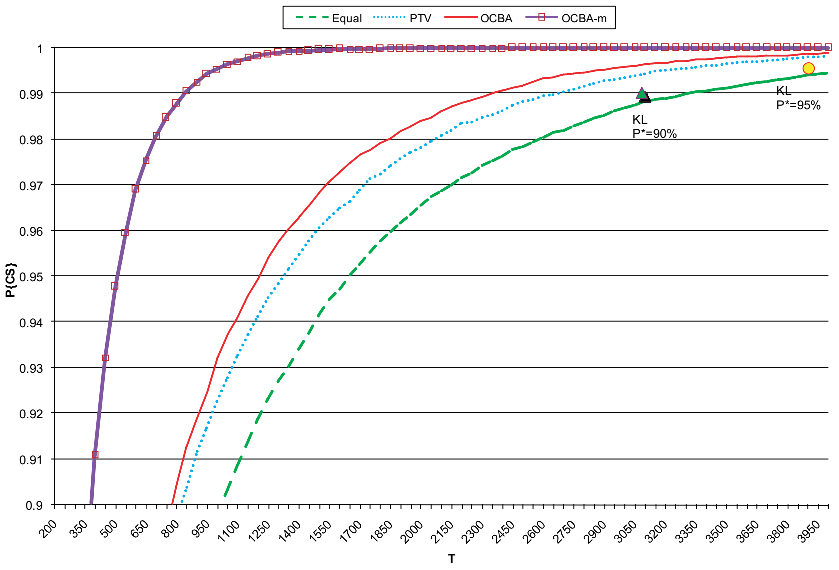


Figure 5.4(a). $P\{\text{CS}\}$ versus T using four sequential allocation procedures and KL (triangle for $P^* = 90\%$ and circle for $P^* = 95\%$) for Experiment 5.2.

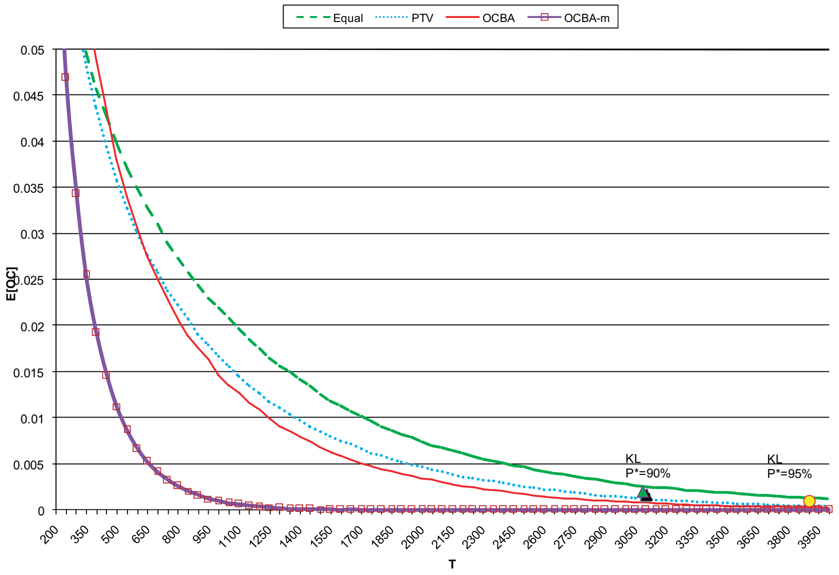


Figure 5.4(b). E[OC] versus T using four sequential allocation procedures and KL (triangle for $P^* = 90\%$ and circle for $P^* = 95\%$) for Experiment 5.2.

For KL, we also test two cases $P^* = 0.9$ and $P^* = 0.95$, and the corresponding estimated $P\{\text{CS}\}$ and E[OC] versus the average total simulation replications are shown as two single points (the triangle and circle) in Figure 5.4. Not surprisingly, the performance of KL is along the performance curve of PTV, since KL basically allocates the computing budget based on design variance. However, KL achieves a substantially higher $P\{\text{CS}\}$ than the desired level (e.g., about 0.99 for the target minimum of $P^* = 0.9$) by spending a much higher computing budget than actually needed.

Experiment 5.3. Generic case: Monotone mean and equal variance

This numerical test is based on the simple generic problem presented in Experiment 4.2.1. There are 10 alternative designs, with distribution $N(i, 6^2)$ for design $i = 1, 2, \ldots, 10$. The goal is to identify the top-3 designs via simulation samples, i.e., $m = 3$ in this experiment.

Figure 5.5 depicts the simulation results. As in earlier experiments, OCBA-m achieves the highest $P\{\text{CS}\}$ and the lowest E[OC]

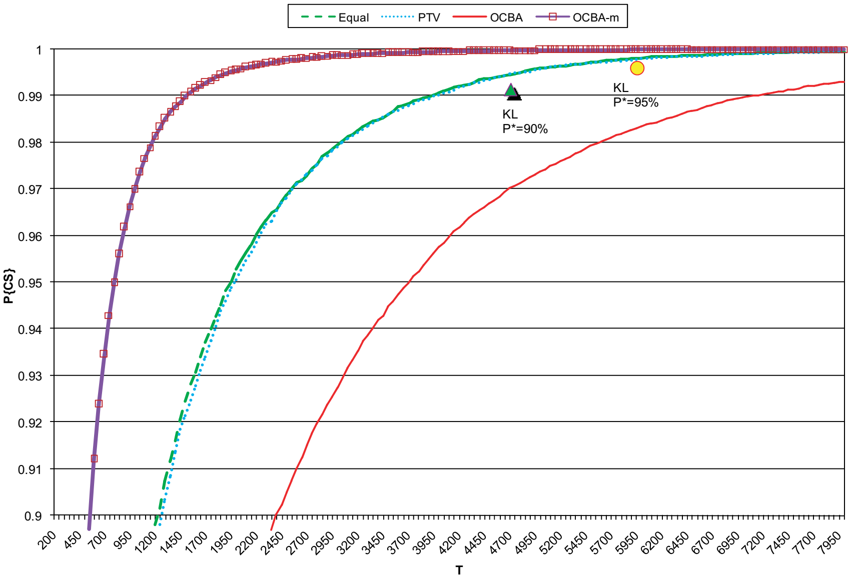


Figure 5.5(a). $P\{\text{CS}\}$ versus T using four sequential allocation procedures and KL (triangle for $P^* = 90\%$ and circle for $P^* = 95\%$) for Experiment 5.3.

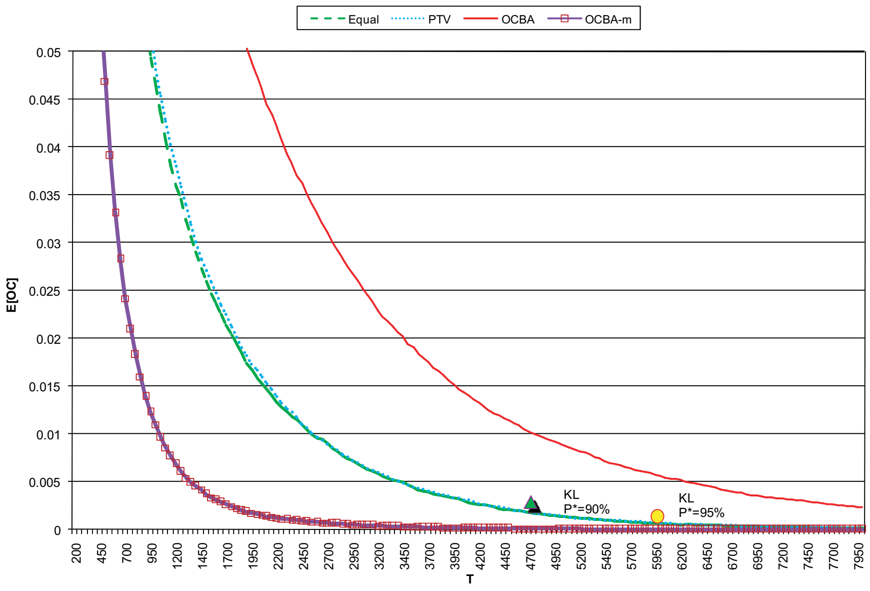


Figure 5.5(b). E[OC] versus T using four sequential allocation procedures and KL (triangle for $P^* = 90\%$ and circle for $P^* = 95\%$) for Experiment 5.3.

with the same amount of computing budget. Specifically, the computation costs to attain $P\{\text{CS}\} = 0.95$ for OCBA-m, OCBA, Equal, and PTV are 800, 3200, 1950, 2000, respectively. Again OCBA-m is significantly faster than other methods and KL achieves a substantially higher $P\{\text{CS}\}$ than the desired level by spending a much higher computing budget than actually needed.

Experiment 5.4. Larger-scale problem

This is a variant of Experiment 5.3, with the number of designs increased to 50. The alternatives have distribution $N(i, 10^2)$ for design $i = 1, 2, \ldots, 50$, and $m = 5$. Since KL's performance basically follows that of PTV, but its required computing budget is far beyond the range we are considering here, we exclude KL from the numerical testing.

Figure 5.6 depicts the simulation results. As in earlier experiments, OCBA-m achieves the highest $P\{\text{CS}\}$ and the lowest

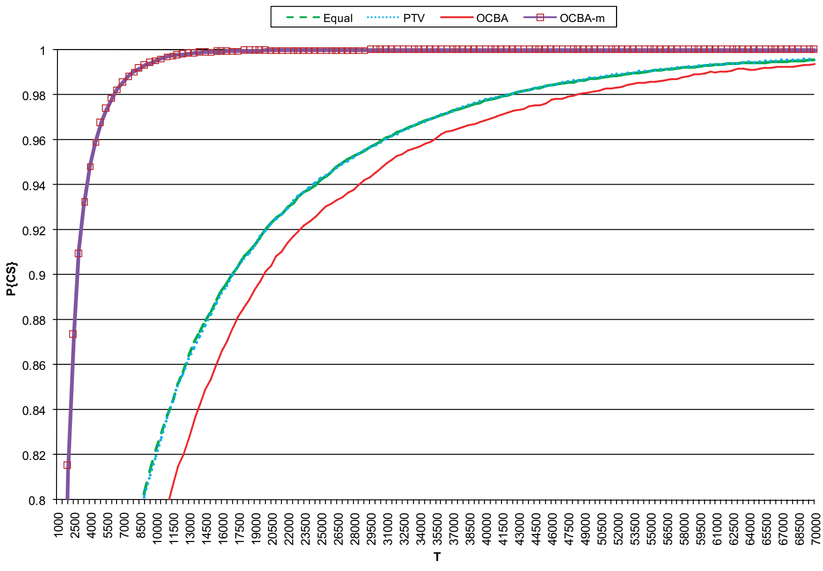


Figure 5.6(a). $P\{\text{CS}\}$ versus T using four sequential allocation procedures for Experiment 5.4.

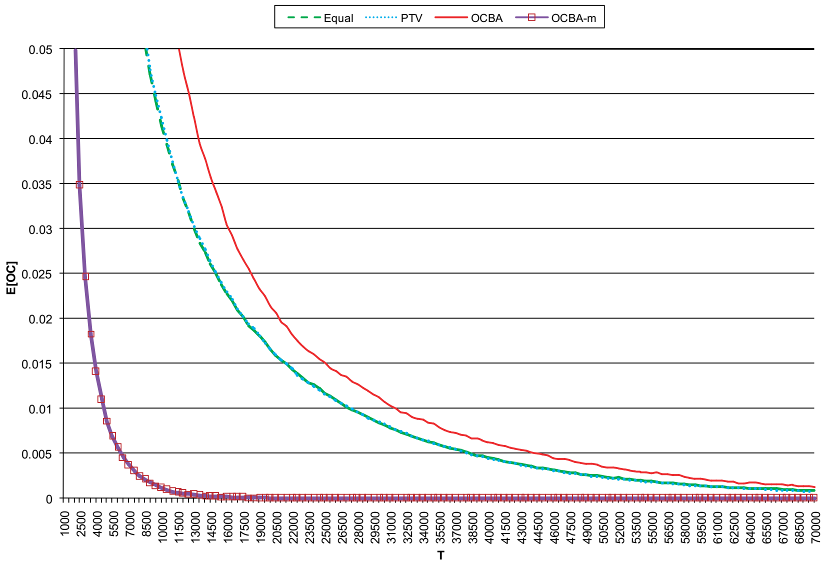


Figure 5.6(b). E[OC] versus T using four sequential allocation procedures for Experiment 5.4.

E[OC] with the same amount of computing budget; however, the performance gap between OCBA-m and other procedures is substantially greater. This is because a larger design space allows the OCBA-m algorithm more flexibility in allocating the computing budget, resulting in even better performance. On the other hand, OCBA performs poorly, because it spends a lot of computing budget on distinguishing the very top designs, a tendency that is penalized even more for larger m. Again, since the variance is constant across designs, the performances of Equal and PTV are nearly indistinguishable. In this experiment, the computation costs to attain $P\{\text{CS}\} = 0.95$ for OCBA-m, OCBA, Equal, and PTV are 4050, 31050, 27050, 27200, respectively.

Chapter 6

Multi-objective Optimal Computing Budget Allocation

In previous chapters, we have developed various OCBA algorithms for allocating simulation runs for identifying the best system design from a set of competitive designs for single objective problems. In this chapter, we will look into the problem where there is more than one objective.

Similar to the previous chapters, we consider a finite design set with k being relatively small, so that we can run simulations for all the designs in order to find the optimal solutions.

Multi-objective optimization, also known as multi-criteria or multi-attribute optimization, is the process of simultaneously optimizing two or more conflicting objectives subject to certain constraints. Without loss of generality, a minimization problem is considered in this chapter. The multi-objective optimization problem can be defined as

$$\min_{\theta \in \Theta} \langle J_1(\theta), \ldots, J_H(\theta) \rangle, \tag{6.1}$$

where Θ is the design space and $(J_1(\theta), \ldots, J_H(\theta))$ are the H objectives for design θ which are to be optimized and $J_l(\theta) = E[L_l(\theta, \omega)]$ is the expected value for the l-th objective which can be estimated from simulation output $L_l(\theta, \omega)$. To simplify the discussion, we denote the design space $\Theta = \{1, \ldots, k\}$, and J_{il} as the l-th objective for design i, i.e., $J_{il} \equiv J_l(\theta_i)$.

In the following sections, we will first introduce the concept of Pareto optimality for the multi-objective optimization problem and based on this concept, we will then present a framework for determining the computing budget allocation rule for the multi-objective problem when performances are evaluated using simulation.

6.1. Pareto Optimality

In multi-objective optimization problems, we might not be able to find a single best solution (or design) that simultaneously optimizes all the objectives. In this case, we may want to find a solution for which each objective has been optimized to the extent that if we try to optimize it any further, then the other objective(s) will become worse. Finding such a solution is a typical goal when setting up and solving a multi-objective optimization problem.

To solve this problem, one of the common approaches is to convert the multi-objective problem into a single objective one. The basic idea is to combine all the objectives by using a functional form, i.e.,

$$J(\theta) = f(J_i(\theta), \ldots, J_H(\theta)). \tag{6.2}$$

One of the most common functional forms used is the linear function, i.e., $J(\theta) = \sum_{i=1}^{H} w_i J_i(\theta)$, where w_i is the weight for the i-th objective. In such a case, we need to specify the scalar weight for each objective, and then the solution that optimizes this combined objective can be found by using any single-objective optimizer. For the stochastic case, the OCBA and OCBA-m approaches which are discussed in Chapter 3 and 5 can be applied directly.

Besides the linear function, another common functional form is multi-attribute utility function (Butler *et al.* 2001). Under such a case, $J(\theta) = \sum_{i=1}^{H} w_i u_i(J_i(\theta))$, where $u_i(.)$ is the utility function for the i-th performance. Similarly OCBA and OCBA-m can be used directly.

No matter which methods are used, the solution obtained will highly depend on the values of the weights and the utility function chosen. However, in actual application, it might not be trivial to determine weights or select a suitable function, especially if the decision analyst is not the one who is making the decision.

Hence, another approach to solve the multi-objective optimization problem is to use the concept of Pareto optimality, where the solution for the problem is a complete set of non-dominated solutions, which is also known as the Pareto set.

The concept of Pareto optimality was first employed to find the non-dominated Pareto set (Fonseca and Fleming 1995) for multi-objective optimization problems. After that, several researchers (Deb 2001, Deb *et al.* 2002) contributed to and enriched the theory of this field.

To find the Pareto set, we first need to define the dominance relationship. A design i is said to be dominated by design j, denoted as $j \prec i$, when design j is not worse than design i for all the performances, and there exist at least one performance that design j is better than design i. The dominance relationship can be defined below.

$$\begin{aligned} j \prec i \text{ iff } \forall l \in \{1, 2, \ldots, H\}, J_{jl} \leq J_{il} \text{ and} \\ \exists l \in \{1, 2, \ldots, H\}, J_{jl} < J_{il}. \end{aligned} \tag{6.3}$$

The design i is not dominated by design j, denoted as $j \not\prec i$, when conditions listed in Equation (6.3) do not hold. A design i is called a non-dominated design in the design space Θ when there are no designs in the Θ dominating design i. A Pareto set consists of all the non-dominated designs in the design space Θ. Specifically, a Pareto set S_p is defined as follows.

$$S_p = \{i | i \in \Theta \text{ and } j \not\prec i, \forall j \in \Theta\}. \tag{6.4}$$

Figure 6.1(a) illustrates the dominance relationship between solutions for the problem with two objectives. Let us denote solution A as our reference solution, and the regions highlighted in different shades of grey in Figure 6.1(a) represent the three different dominance relations with design A. Designs located in the dark grey region are dominated by design A because A is better in both objectives. For the same reason, designs located in the white region dominate design A. As for those designs in the light grey region, they are neither dominating nor being dominated by design A. However, for those designs

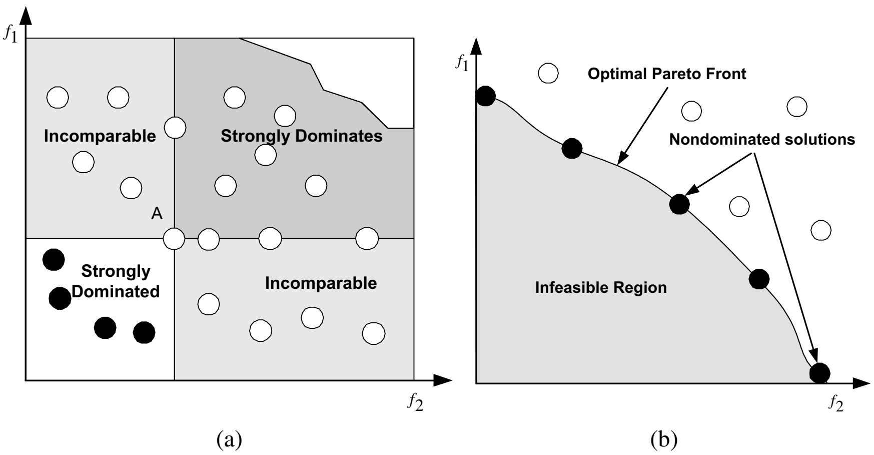


Figure 6.1. Illustration of the (a) Pareto dominance relationship between candidate designs relative to design A and (b) Non-dominated designs and Pareto designs.

which are located at the boundaries between the dark and light grey regions, they are dominated by A because A has one better objective when compared to these designs.

Non-dominated designs are those solutions in which we cannot find any feasible designs that dominate them, and they are shown in Figure 6.1(b). These non-dominated designs will form an optimal Pareto front. Pareto set is defined as the set which consists of all the non-dominated designs.

6.2. Multi-objective Optimal Computing Budget Allocation Problem

When the performance of a multi-objective optimization problem is estimated using simulation, the problem is known as the multi-objective simulation optimization problem. Under this case, we want to determine the optimal allocation rule in such a way that we can select the correct Pareto set with a high probability. The derivation of the allocation rule will be similar to OCBA for the single objective case presented in Chapter 3, except that the concept of

Pareto optimality will be incorporated into the OCBA framework. In the following subsections, we will explain the concept and derive the new allocation rule for multi-objective problems.

6.2.1. *Performance index for measuring the dominance relationships and the quality of the selected Pareto set*

Denote

$\boldsymbol{J}_i$: The vector of the performance measures of design i; i.e., $\boldsymbol{J}_i = (J_{i1}, J_{i2}, \ldots, J_{iH})$

$\tilde{\boldsymbol{J}}_i$: A vector of random variables which have the posterior distributions of the performance measures of design i; $\tilde{\boldsymbol{J}}_i = (\tilde{J}_{i1}, \tilde{J}_{i2}, \ldots, \tilde{J}_{iH})$.

$\boldsymbol{X}_i$: A matrix that contains the simulation output for design i. $\boldsymbol{X}_i = (X_{iln})$ where X_{iln} is the l-th objective of design i for simulation replication n.

To make the derivation for the allocation rule more tractable, we assume that simulation outputs are independently distributed across: 1) different replications; 2) different designs; and 3) different performance measures of the same design. Therefore $X_{iln}$is independent with each other. The optimal simulation allocation rule will be derived based on the Bayesian framework presented in Chapter 3.

For design i, the performance measures $\boldsymbol{J}_i$ are unknown, and can only be estimated using the simulation outputs $\boldsymbol{X}_i$. Assume that the performance measure J_{il}, for $l = 1, 2, \ldots, H$, has a conjugate normal prior distribution. By considering non-informative prior distributions, and given that $X_{il1}, X_{il2}, \ldots, X_{ilN_i}$ are N_i independent simulation observations for the l-th objective of design i, and σ_{il}^2 is the known variance of the l-th objective of design i, then according to Chapter 3, the unknown performance measure J_{il} can be described by the following posterior distribution,

$$\tilde{J}_{il} \sim N\left(\bar{J}_{il}, \frac{\sigma_{il}^2}{N_i}\right), \tag{6.5}$$

where $\bar{J}_{il} = \frac{1}{N_i}\sum_{n=1}^{N_i} X_{iln}$ is the sample mean of the simulation output for the l-th objective of design i, and σ_{il}^2 is assumed to be known, but in the actual application can be approximated by the sample variance $\hat{\sigma}_{il}^2 = \frac{1}{N_i-1}\sum_{n=1}^{N_i}(X_{iln} - \bar{J}_{il})^2$. (For ease of notation, we will use σ_{il}^2 in our derivation below.)

In addition to using the Bayesian framework to estimate the posterior distribution for performance measure $\tilde{J}_{il}$, we use it to estimate the predictive posterior distribution of $\tilde{J}_{il}$ if additional replications are allocated to design i. This is extensively discussed in Section 3.6. Suppose we have already obtained N_i independent simulation observations $X_{il1}, X_{il2}, \ldots, X_{ilN_i}$; if additional Δ_i replications are to be allocated to design i, we can approximate the posterior distribution for $\tilde{J}_{il}$ (without actually running the Δ_i replications) by

$$\tilde{J}_{il} \sim N\left(\bar{J}_{il}, \frac{\sigma_{il}^2}{N_i + \Delta_i}\right). \tag{6.6}$$

6.2.1.1. *A performance index to measure the degree of non-dominated for a design*

Under the concept of Pareto optimality, the fitness of a design is measured in terms of its dominance relationship. As there is simulation noise involved in the performance measures, we need to define the probability that design j dominates design i, which is denoted by $P(j \prec i)$. Equation (6.7) defines how to compute $P(j \prec i)$ given the initial simulation output $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$.

$$P(j \prec i) = P(\tilde{J}_j \prec \tilde{J}_i|\boldsymbol{X}_i, \boldsymbol{X}_j) = \prod_{l=1}^{H} P(\tilde{J}_{jl} \le \tilde{J}_{il}|\boldsymbol{X}_i, \boldsymbol{X}_j). \tag{6.7}$$

To simplify the notations, we use $P(\tilde{J}_j \prec \tilde{J}_i)$ and $P(\tilde{J}_{jl} \le \tilde{J}_{il})$ to represent $P(\tilde{J}_j \prec \tilde{J}_i|\boldsymbol{X}_i, \boldsymbol{X}_j)$ and $P(\tilde{J}_{jl} \le \tilde{J}_{il}|\boldsymbol{X}_i, \boldsymbol{X}_j)$, respectively, in the remainder of this chapter.

For a set of k designs, the following performance index ψ_i measures how non-dominated a design i is

$$\psi_i = P\left(\bigcap_{\substack{j\in\Theta \\ j\neq i}} (\tilde{J}_j \not\prec \tilde{J}_i)\right), \tag{6.8}$$

where $\tilde{J}_j \not\prec \tilde{J}_i$ denotes that design i is not dominated by design j. The index ψ_i measures the probability that design i is not dominated by all the designs in the design space Θ. When there is no simulation noise (i.e., the performance measure is deterministic), a design will be called a non-dominated design when $\psi_i = 1$. As it is difficult to accurately compute ψ_i for stochastic problems, we find an upper and lower bound for it, as stated in the following lemma.

Lemma 6.1. *The performance index ψ_i can be bounded as follows.*

$$\prod_{\substack{j\in\Theta\\ j\neq i}}\left[1-\prod_{l=1}^{H}P(\tilde{J}_{jl}\le \tilde{J}_{il})\right] \le \psi_i \le \min_{\substack{j\in\Theta\\ j\neq i}}\left[1-\prod_{l=1}^{H}P(\tilde{J}_{jl}\le \tilde{J}_{il})\right]. \tag{6.9}$$

Proof. See Appendix C.1. □

6.2.1.2. *Construction of the observed Pareto set*

We will select the non-dominated designs amongst the k designs to form the Pareto set. As the performance measure is unknown, we can only select the designs based on the sample means, and so we call this selected set the observed Pareto set.

Given that $\bar{J}_{il}$ is the sample mean of the l-th objective of design i, design j dominates design i by observation, which is denoted by $j\hat{\prec}i$, if the following condition holds with at least one inequality being strict:

$$\bar{J}_{jl} \le \bar{J}_{il} \quad \text{for } l = 1, 2, \ldots, H. \tag{6.10}$$

The observed Pareto set consists of all the designs that we cannot find any other designs dominating them in observation. The designs that are not in the observed Pareto set will be in the observed non-Pareto set. According to the definition of performance index ψ_i, at the end of the simulation, with enough simulation budget, ψ_i should approach 1 for designs in the observed Pareto set, whereas for designs in the observed non-Pareto set, ψ_i should approach 0.

6.2.1.3. *Evaluation of the observed Pareto set by two types of errors*

A good observed Pareto set should have the following two properties: i) The designs in the observed Pareto set are actually the non-dominated designs with high probability; and ii) the designs in the observed non-Pareto set are actually dominated by some designs with high probability. Specifically, we evaluate the quality of an observed Pareto set by measuring its Type I and Type II errors.

The following notations will be used in the definition of the Type I and Type II errors:

S_p: The observed Pareto set.
$\bar{S}_p$: The observed non-Pareto set, $\bar{S}_p = \Theta \backslash S_p$.
E_i: The event that design i is non-dominated by all other designs.
E_i^c: The event that design i is dominated by at least one design.

From Equation (6.8), we know that $P\{E_i\} = \psi_i$ and $P\{E_i^c\} = 1 - \psi_i$.

Type I error:

Type I error occurs when some designs that are actually non-dominated are in the observed non-Pareto set. It is the probability that at least one design in the observed non-Pareto set is non-dominated. The probability that all designs in the observed non-Pareto set are dominated by at least one other design is:

$$P\left\{ \bigcap_{i \in \bar{S}_p} E_i^c \right\}. \tag{6.11}$$

Therefore, Type I error denoted by e_1 is:

$$e_1 = 1 - P\left\{ \bigcap_{i \in \bar{S}_p} E_i^c \right\}. \tag{6.12}$$

Type II error:

Type II error occurs when some designs that are actually dominated by other designs are included in the observed Pareto set. It is the probability that at least one design in the observed Pareto set is actually dominated by another design. The probability that the designs in the observed Pareto set are all non-dominated is:

$$P\left\{ \bigcap_{i \in S_p} E_i \right\}. \tag{6.13}$$

So the Type II error, which is denoted by e_2, is:

$$e_2 = 1 - P\left\{ \bigcap_{i \in S_p} E_i \right\}. \tag{6.14}$$

When both types of errors approach 0, all designs in the observed Pareto set will be non-dominated, and all designs in the observed non-Pareto set will be dominated by some designs. In this case, we can say that the true Pareto set is correctly found.

The following lemma provides approximations of both types of errors. They are Approximated Type I error ((ae_1) and Approximated Type II error (ae_2), respectively.

Lemma 6.2. *The Type I and Type II errors can be bounded from above by the following ae_1 and ae_2, respectively:*

$$e_1 \le ae_1 = \sum_{i \in \bar{S}_p} \psi_i, \tag{6.15}$$

$$e_2 \le ae_2 = \sum_{i \in S_p} (1 - \psi_i). \tag{6.16}$$

Proof. See Appendix C.2.

ae_1 can be interpreted as the sum of the non-dominated probabilities for all the designs in non-Pareto set. It will be zero when all the designs in the non-Pareto set are dominated by some designs. Similarly, ae_2 can be interpreted as the sum of the dominated probabilities for all the designs in the Pareto set. It will be equal to zero when all the designs in the Pareto set are non-dominated.

As ψ_i cannot be explicitly expressed, we establish other upper bounds for both ae_1 and ae_2 in the following lemma.

Lemma 6.3. *We have the following valid upper bounds for Type I and Type II errors*:

$$e_1 \le ae_1 \le ub_1 = H|\bar{S}_p| - H \sum_{i \in \bar{S}_p} \max_{\substack{j \in \Theta \\ j \ne i}} \left[\min_{l \in \{1,\ldots,H\}} P(\tilde{J}_{jl} \le \tilde{J}_{il}) \right], \quad (6.17)$$

$$e_2 \le ae_2 \le ub_2 = (k-1) \sum_{i \in S_p} \max_{\substack{j \in \Theta \\ j \ne i}} \left[\min_{l \in \{1,\ldots,H\}} (P(\tilde{J}_{jl} \le \tilde{J}_{il})) \right]. \quad (6.18)$$

Proof. See Appendix C.3.

Let j_i denote the design that dominates design i with the highest probability, and $l^i_{j_i}$ denote the objective of j_i that dominates the corresponding objective of design i with the lowest probability, i.e.,

$$j_i = \arg\max_{\substack{j \in S \\ j \ne i}} \prod_{l=1}^{H} P(\tilde{J}_{jl} \le \tilde{J}_{il}), \quad (6.19)$$

$$l^i_{j_i} = \arg \min_{l \in \{1,\ldots,H\}} P(\tilde{J}_{j_i l} \le \tilde{J}_{il}). \quad (6.20)$$

Then the two upper bounds can be expressed as:

$$e_1 \le ae_1 \le ub_1 = H|\bar{S}_p| - H \sum_{i \in \bar{S}_p} P(\tilde{J}_{j_i l^i_{j_i}} \le \tilde{J}_{i l^i_{j_i}}), \quad (6.21)$$

$$e_2 \le ae_2 \le ub_2 = (k-1) \sum_{i \in S_p} P(\tilde{J}_{j_i l^i_{j_i}} \le \tilde{J}_{i l^i_{j_i}}). \quad (6.22)$$

The interpretation for the bounds is as follows. Consider a design i in the non-Pareto set, as we want to show that it is dominated by some designs. Thus we need to find a design that dominates it with the highest probability, and this design is design j_i. After that, we only consider the objective that j_i is better than design i by the least amount, and this objective is the $l^i_{j_i}$-th objective. If the probability that design j_i is better than design i for the $l^i_{j_i}$-th objective approaches 1, then we will be sure that design i belongs to the non-Pareto set as it is dominated by at least one design, i.e., design j_i.

Similarly, consider a design i in the Pareto set, as we want to show that this design is non-dominated. Thus we need to find a design that dominates it the most, i.e., design j_i. Then, we will consider the objective that the design j_i is worse than the design i the most, and this objective is the $l^i_{j_i}$-th objective. When this probability goes to zero, we will be sure that the design that dominates design i with the highest probability actually does not dominate design i due to the fact that for the $l^i_{j_i}$-th objective, design i is better than design j_i. Hence design i is non-dominated.

6.2.2. *Formulation for the multi-objective optimal computing budget allocation problem*

To determine the computing budget allocation rule, we need to define the objectives for the overall problem. Similar to the Chapter 3, we can use the probability of correct selection as the main objective. However, we want to select a set instead of a single design, so the probability of correct selection $P\{\text{CS}\}$ is defined as the probability that the observed Pareto set is the true Pareto set. Specifically,

$$P\{\text{CS}\} \equiv P\left\{\left(\bigcap_{i\in S_p} E_i\right)\bigcap\left(\bigcap_{i\in \bar{S}_p} E_i^c\right)\right\}. \tag{6.23}$$

It is the probability that all the designs in the Pareto set are non-dominated, and all the designs in the non-Pareto set are dominated by at least one design.

To maximize simulation efficiency, we want to determine the best numbers of simulation replications for each design so as to maximize the probability of correct selection. Specifically, we wish to choose $N_1, N_2, \ldots, N_k$ such that $P\{\text{CS}\}$ is maximized, subject to a limited computing budget T, i.e.,

(MOCBA-PCS Problem)

$$\max_{N_1,\ldots,N_k} \; P\{\text{CS}\}$$

$$\text{s.t. } N_1 + N_2 + \cdots + N_k = T \text{ and } N_i \geq 0. \tag{6.24}$$

6.3. Asymptotic Allocation Rule

Similar to the single objective case, there is no explicit equation to compute $P\{\text{CS}\}$. Monte Carlo simulation can be applied to estimate this probability. However, as we need to derive an allocation rule explicitly, we use the Approximate Probability of Correct Selection *APCS-M* to approximate $P\{\text{CS}\}$. In fact *APCS-M* is a lower bound for $P\{\text{CS}\}$.

Lemma 6.4. *APCS-M is the lower bound for* $P\{\text{CS}\}$, *where*

$$\textit{APCS-M} \equiv 1 - ub_1 - ub_2. \tag{6.25}$$

Proof.

$$\text{P}\{\text{CS}\} \geq 1 - \left\{1 - P\left(\bigcap_{i \in S_p} E_i\right)\right\} - \left\{1 - P\left(\bigcap_{i \in \bar{S}_p} E_i^c\right)\right\} = 1 - e_1 - e_2.$$

Hence,

$$\text{P}\{\text{CS}\} \geq 1 - ub_1 - ub_2 = \textit{APCS-M}. \qquad \square$$

Using the approximation given by Lemma 6.4, we consider the following approximate OCBA problem:

(MOCBA-APCS-M Problem)

$$\max_{N_1,\ldots,N_k} \ \textit{APCS-M}$$

$$\text{s.t. } N_1 + N_2 + \cdots + N_k = T \text{ and } N_i \geq 0. \tag{6.26}$$

To derive the asymptotic allocation rules, we adopt the similar approach in Chapter 3. Specifically, we apply the Lagrange method and assume $T \to \infty$. Define α_i as the proportion of the total computing budget allocated to design i. Thus, $N_i = \alpha_i T$, and $\sum_{i=1}^{k} \alpha_i = 1$. The asymptotic allocation rules obtained are given in Lemma 6.5.

Lemma 6.5. *As $T \to \infty$, the APCS-M can be asymptotically maximized when*:

$$\textit{For designs } h \in S_A,\ \beta_h = \frac{\left(\hat{\sigma}^2_{hl^h_{j_h}} + \hat{\sigma}^2_{j_h l^h_{j_h}}/\rho_h\right)/\delta^2_{hj_h l^h_{j_h}}}{\left(\hat{\sigma}^2_{ml^m_{j_m}} + \hat{\sigma}^2_{j_m l^m_{j_m}}/\rho_m\right)/\delta^2_{mj_m l^m_{j_m}}}. \tag{6.27}$$

$$\textit{For a design } d \in S_B, \beta_d = \sqrt{\sum_{i \in \Theta_d^*} \frac{\sigma^2_{dl_d^i}}{\sigma^2_{il_d^i}} \beta_i^2}. \tag{6.28}$$

Then the allocation quantity α_i will be

$$\alpha_i = \frac{\beta_i}{\sum_{j \in \Theta} \beta_j}, \tag{6.29}$$

where

$$\delta_{ijl} = \overline{J}_{jl} - \overline{J}_{il}, \tag{6.30}$$

$$j_i \equiv \arg \max_{\substack{j \in \Theta \\ j \neq i}} \prod_{l=1}^{H} P(\tilde{J}_{jl} \leq \tilde{J}_{il}), \tag{6.31}$$

$$l_{j_i}^i \equiv \arg \min_{l \in \{1,\ldots,H\}} P(\tilde{J}_{j_i l} \leq \tilde{J}_{il}), \tag{6.32}$$

$$S_A \equiv \left\{ \textit{design } h \in S \;\middle|\; \frac{\delta^2_{hj_h l_{j_h}^h}}{\frac{\hat{\sigma}^2_{hl_{j_h}^h}}{\alpha_h} + \frac{\hat{\sigma}^2_{j_h l_{j_h}^h}}{\alpha_{j_h}}} < \min_{i \in \Theta_h} \frac{\delta^2_{ihl_h^i}}{\frac{\hat{\sigma}^2_{il_h^i}}{\alpha_i} + \frac{\hat{\sigma}^2_{hl_h^i}}{\alpha_h}} \right\}, \tag{6.33}$$

$$S_B \equiv S \backslash S_A, \tag{6.34}$$

$$\Theta_h = \{i | i \in S, j_i = h\}, \tag{6.35}$$

$$\Theta_d^* = \{h | h \in S_A, j_h = d\}, \tag{6.36}$$

$$\rho_i = \alpha_{j_i} / \alpha_i. \tag{6.37}$$

Proof. See Appendix C.4. □

Though the rule looks quite complicated, it can actually be classified into two major types of rule, depending on the role each design is playing.

(i) For the designs which play the role of being dominated (i.e., they are dominated by other designs), the allocation rule will be the ratio rule (i.e., the allocation quantity is proportional to the noise to signal ratio).

(ii) For those designs which play the role of dominating, the allocation rule will be the squares root rule (i.e., the allocation quantity for the design will be the square root for the sum of weighted squares for those designs that it dominates).

This is quite similar to the OCBA results, where the best design is using the square root rule, while the other designs are using the ratio rule. However, in the multi-objective problem, as more than one designs are playing the dominating roles, and it might also be possible that a design is playing both roles (due to the stochastic nature, a design might be dominated by one design with high probability, but also dominating others with high probability), so we need to have some rules to determine which roles the designs are playing.

In general, the designs in S_A play the role of being dominated, and designs in S_B play the role of dominating. To determine which designs are in these sets, we need to use Equation (6.33). The left side of the inequality for these two equations is the signal to noise ratio for the design to play the role for being dominated, and the right-hand side of the inequality is the signal to noise ratio when the design plays the role of dominating. The smaller the signal to noise ratio, the more critical a role this design plays in determining the probability of correct selection. In order to maximize the probability of correct selection, we improve the simulation precision for those critical designs with the smaller signal to noise ratios (more computing budget should be allocated to them).

It is not trivial to compute the allocation quantity given in Lemma 6.5 because, to compute Equations (6.27) to (6.29), we need to know the values from Equations (6.30) to (6.37). However, the values from Equations (6.30) to (6.37) also depends on the value of α_i. One possible numerical method is to solve this problem iteratively to search for α_i. We first assume initial values for α_i, and then solve Equations (6.30) to (6.37). After computing the new values for α_i, we repeat the process until α_i converges.

For ease of computation, we offer a simplified version of allocation rule summarized as follows.

Lemma 6.6. *The asymptotic rule given in Lemma 6.5 can be approximated as follows.*

$$\text{For } h, m \in S_A, \quad \frac{\alpha_h}{\alpha_m} = \left(\frac{\sigma_{hl^h_{j_h}} / \delta_{hj_h l^h_{j_h}}}{\sigma_{ml^m_{j_m}} / \delta_{mj_m l^m_{j_m}}} \right)^2. \tag{6.38}$$

$$\text{For } d \in S_B, \quad \alpha_d^2 = \sum_{h \in \Theta_d^*} \frac{\sigma^2_{dl^h_d}}{\sigma^2_{hl^h_d}} \alpha_h^2, \tag{6.39}$$

where

$$\delta_{ijl} = \overline{J}_{jl} - \overline{J}_{il}, \tag{6.40}$$

$$l^i_j \equiv \arg \min_{l \in \{1,\ldots,H\}} P(\tilde{J}_{jl} \leq \tilde{J}_{il}) = \arg \max_{l \in \{1,\ldots,H\}} \frac{\delta_{ijl}|\delta_{ijl}|}{\sigma^2_{il} + \sigma^2_{jl}}, \tag{6.41}$$

$$j_i \equiv \arg \max_{\substack{j \in \Theta \\ j \neq i}} \prod_{l=1}^{H} P(\tilde{J}_{jl} \leq \tilde{J}_{il}) = \arg \min_{\substack{j \in \Theta \\ j \neq i}} \frac{\delta_{ijl^i_j}|\delta_{ijl^i_j}|}{\sigma^2_{il^i_j} + \sigma^2_{jl^i_j}}, \tag{6.42}$$

$$S_A = \left\{ h | h \in S, \frac{\delta^2_{hj_h l^h_{j_h}}}{\sigma^2_{hl^h_{j_h}} + \sigma^2_{j_h l^h_{j_h}}} < \min_{i \in \Theta_h} \frac{\delta^2_{ihl^i_h}}{\sigma^2_{il^i_h} + \sigma^2_{hl^i_h}} \right\}, \tag{6.43}$$

$$S_B = S \backslash S_A, \tag{6.44}$$

$$\Theta_h = \{i | i \in S, j_i = h\}, \quad \Theta_d^* = \{h | h \in S_A, j_h = d\}. \tag{6.45}$$

In the simplified rule, after obtaining the sample means and sample variances from the initial simulation runs, we can evaluate Equations (6.40) to (6.45) to determine the sets S_A and S_B. Then we can compute the allocation quantity from Equations (6.38) and (6.39) directly. Iterative search is no longer needed.

The simplification is made based on some approximations. First we ignore the impact of the allocation quantity in determining the role of dominating or being dominated. Also in computing the allocation quantity, we use the variance associated with each design directly.

6.4. A Sequential Allocation Procedure

In this section, we propose a heuristic sequential procedure to implement the MOCBA algorithm (or the simplified MOCBA algorithm) and it is shown as follows:

MOCBA Procedure (Maximizing APCS-M)

```
INPUT          $k$, $T_{\max}$, $\delta$, $\Delta$, $n_0$ ($T_{\max} - kn_0$ is a multiple of
               $\Delta$ and $n_0 \geq 5$);
INITIALIZE     $l \leftarrow 0$;
               Perform $n_0$ simulation replications for all
               designs; $N_1^l = N_2^l = \cdots = N_k^l = n_0$.
               $T_l = kn_0$
LOOP           WHILE $T_l < T_{\max}$ DO
    UPDATE     Calculate sample means, and sample standard
               deviation using the new simulation output;
               Construct the observed Pareto set $S_p$;
    ALLOCATE   Increase the computing budget by $\Delta$, i.e.,
               $T_{l+1} = T_l + \Delta$ and calculate the new budget
               allocation $\alpha_1^{l+1}, \alpha_2^{l+1}, \ldots, \alpha_k^{l+1}$, according to
               Equations (6.27) to (6.37) if it is the MOCBA
               algorithm, or Equations (6.38) to (6.45) if
               it is the MOCBA simplified algorithm, let
               $N_i^{l+1} = \alpha_i^{l+1} T_{l+1}$;
    SIMULATE   Perform additional $\min(\delta, \max(0, N_i^{v+1} -$
               $N_i^v))$ simulations for design $i$, $i = 1, \ldots, k$;
               $T_{l+1} = \sum_{i=1}^k N_i^l$ and $l \leftarrow l + 1$.
END OF LOOP
```

Note that δ (a fixed given value) is an input parameter which is the maximum number of replications allowed to be allocated to a design at each iteration. The setting of δ can prevent the allocation of too many replications to one design in an iteration possibly caused by numerical approximation error.

Theorem 6.1. *As the simulation budget $T \to \infty$, by following the allocation rules as given in Lemma* 6.5 *and* 6.6, *the observed Pareto* *set in MOCBA approaches the true Pareto set asymptotically with probability when all* $\alpha_i \neq 0$.

Proof. The reason is obvious, because if all the designs will be allocated with non-zero allocation, when $T \to \infty$, there will be no simulation error and so we will be able to get the correct Pareto set. $\square$

6.5. Numerical Results

In this section, we compare the performance of our proposed algorithms (MOCBA) and the simplified one with the equal allocation rule, which simply allocates the same number of simulation replications to each design. The performance indicator used here is the empirical probability of correct selection, which is the percentage of the times that the true Pareto set is selected in observation out of 1000 independent selection experiments.

We test the allocation rules using 3 typical test problems. The first test problem shown in Section 6.4.1 is a simple 3-design case for purpose of easy illustration. The second test problem presented in Section 6.4.2 is a case with designs linearly spread, and Section 6.4.3 gives the third problem which is a general one with multiple designs in the Pareto set. Test problems 2 and 3 are also called neutral case and steep case, respectively, where the concepts of different sets of test problem are introduced in Zhao *et al.* (2005).

6.5.1. *A 3-design case*

In this test case, we consider a problem with 3 designs, where each design is evaluated in 2 objectives. We consider two different scenarios, where scenario I is the case with the same variance for each objective of each design, and scenario II is the case with different variances for individual designs. The true response values for scenario I are listed in Table 6.1. The simulation output of each replication for each design is generated as random numbers from the related normal distribution for each objective. In this scenario, the standard

Table 6.1. Response for the 3-design case I.

Index	Obj. 1	Standard dev. 1	Obj. 2	Standard dev. 2
0	1	5	2	5
1	3	5	1	5
2	5	5	5	5

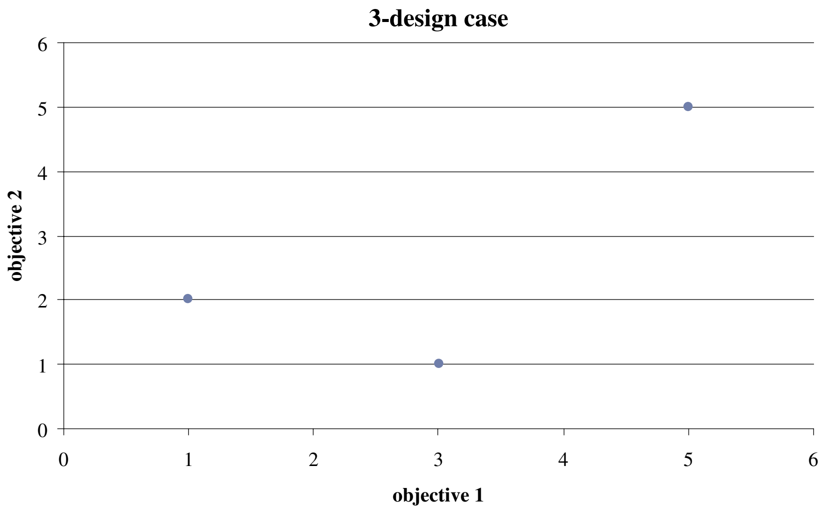


Figure 6.2. Designs spread for the 3-design case I.

deviation is assumed to be the same for each objective of all designs. The designs spread in the design space are shown in Figure 6.2.

It is easily seen that designs 0 and 1 are in the Pareto set, whereas design 2 is not. We conduct 1000 independent experiments for equal allocation, MOCBA and MOCBA_simplified rules. The performance of each allocation rule is tested under an increasing number of total computing budgets, from 50 to 500 with a step increase of 50, and the numerical result is shown in Figure 6.3, where the X-axis denotes an increased number of computing budgets and the Y-axis denotes the corresponding probability of correct selection.

We can see that all the three allocation rules obtain higher probability of correct selection when the computing budget increases. This is intuitive since estimation of ordinal comparisons can be more

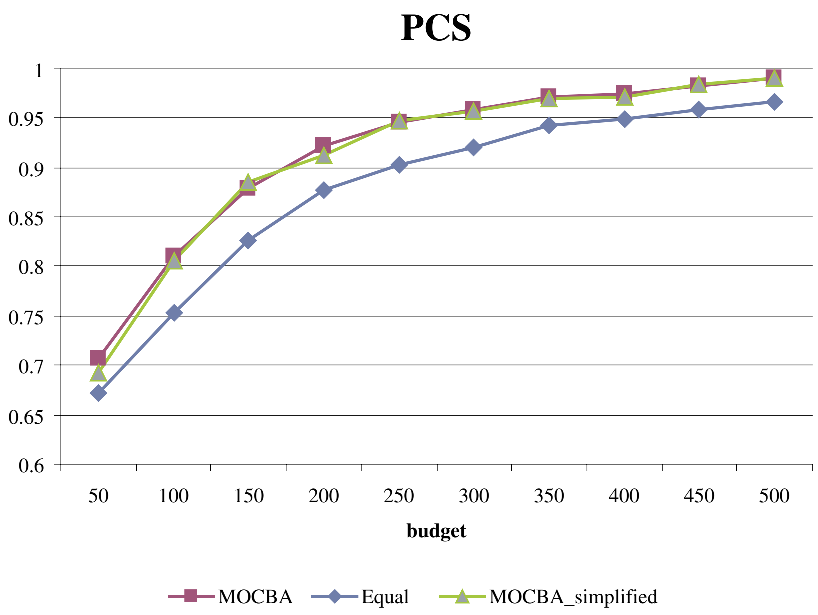


Figure 6.3. Comparison of $P\{\text{CS}\}$ for the 3-design case I.

accurate with more simulation replications. With the same simulation budget, the probability of correct selection of MOCBA and MOCBA_simplified are always higher than that of equal allocation, and converge faster to 1 than equal allocation does. Moreover, the simplified rule performs just as well as the original MOCBA, without showing obvious compromise in performance due to the simplification made in this allocation rule. The comparison can also be made in terms of the total computing budgets to achieve a certain level of $P\{\text{CS}\}$, and the result is shown in Table 6.2. We see that

Table 6.2. Comparison of simulation budgets for the 3-design case I.

$P\{\text{CS}\}$	MOCBA	EQUAL	Saving
0.95	250	400	37.5%
0.96	300	450	33.3%
0.97	350	500	30.0%

to achieve the same level of probability of correct selection, MOCBA always needs fewer simulation budgets than EQUAL ALLOCATION, and the savings are quite substantial. For this specific 3-design case, MOCBA can save as much as 37% of the computational effort compared with equal allocation.

The true response values for the 3-design case with different variance are listed in Table 6.3.

The same simulation procedures are conducted as is done for scenario I and the simulation outputs are illustrated in Figure 6.4 and Table 6.4. The results are similar to that of scenario I, where MOCBA

Table 6.3. Design information for the 3-design case II.

Index	Obj. 1	Standard dev. 1	Obj. 2	Standard dev. 2
1	1	5	2	5
2	3	4	1	4
3	5	3	5	3

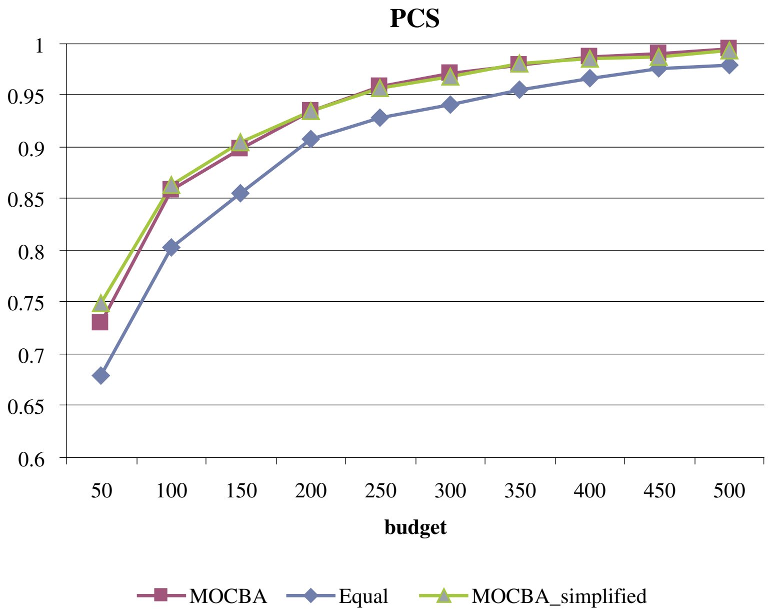


Figure 6.4. Comparison of $P\{\text{CS}\}$ for the 3-design case II.

Table 6.4. Comparison of simulation budgets for the 3-design case II.

$P\{\text{CS}\}$	MOCBA	Equal	Saving
0.96	260	375	30.7%
0.97	290	410	29.3%
0.98	350	500	30.0%

and MOCBA_simplified can achieve a higher $P\{\text{CS}\}$ with a fixed number of simulation budgets. To achieve the same level of $P\{\text{CS}\}$, MOCBA and MOCBA_simplified can always make substantial savings. The results also indicate that the simplified rule can perform as well as the original MOCBA and both of their performances are robust with the variance changes.

6.5.2. *Test problem with neutral spread designs*

In this test case, we consider a problem with 20 designs in total, and they are spread in the design space as shown in Figure 6.5. Thus only the lower left design stays in the Pareto set. The variance for each objective of each design is set as 3^2.

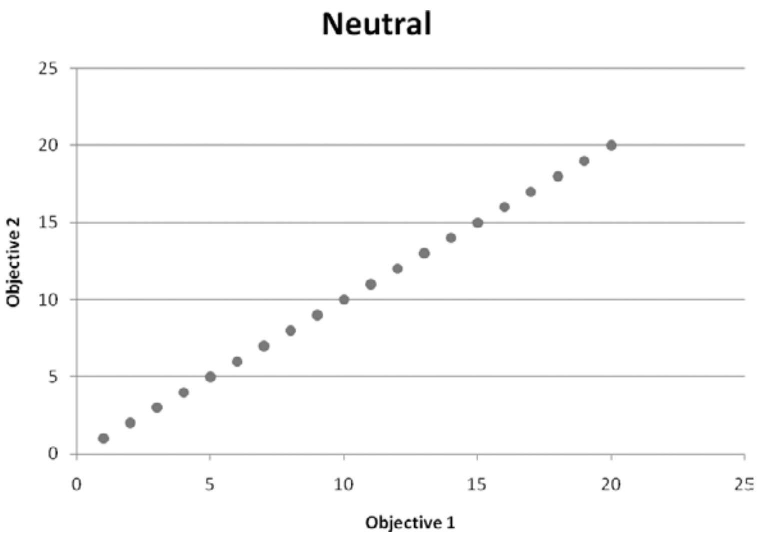


Figure 6.5. Designs with neutral spread.

The total simulation budget is varied from 200 to 2000 with a step increase of 200. The test results shown in Figure 6.6 and Table 6.5 are qualitatively similar to those in previous example. Again MOCBA and MOCBA_simplified perform significantly better than equal allocation. The time saving is higher in this example, partially due to the fact that the best design is so unique and outstanding as compared with other designs, the allocation rule of MOCBA and the simplified one can gain most of allocation advantage whereas equal allocation naively allocates equally without considering the problem structure. The higher efficiency is also due to the larger design space which gives the MOCBA algorithms more flexibility in allocating the computing budget.

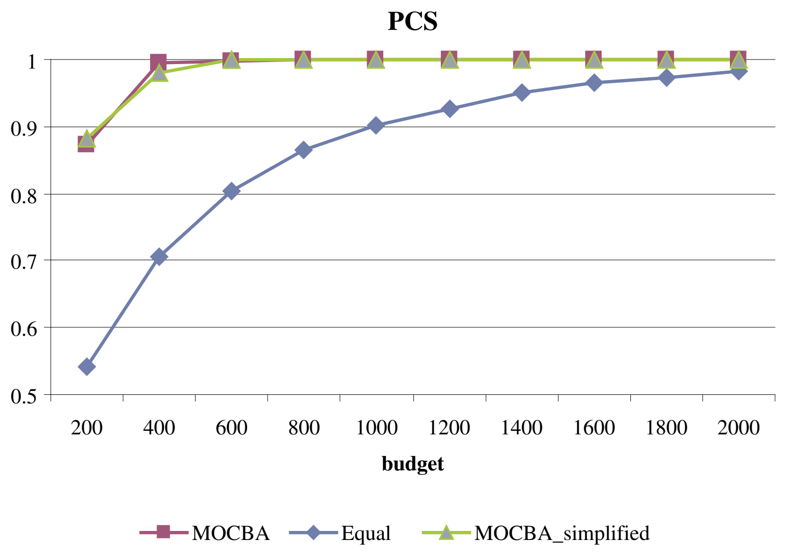


Figure 6.6. Comparison of $P\{\text{CS}\}$ for the neutral case.

Table 6.5. Comparison of simulation budgets for the neutral case.

$P\{\text{CS}\}$	MOCBA	Equal	Saving
0.88	200	900	77.8%
0.99	400	2400	83.3%

6.5.3. *Test problem with steep spread designs*

This is a more general case where there are multiple designs in the Pareto set and the spread of the designs is sparser. There are 16 designs in total and the designs are spread in the design space as illustrated in Figure 6.7. The variance of each objective of each design is set as 2^2.

Simulation experiments similar to previous test cases are conducted and the simulation results in terms of $P\{\text{CS}\}$ and simulation budget savings are shown in Figure 6.8 and Table 6.6, respectively. The simulation budget savings are all compared with equal allocation. Again, all the three allocation rules obtain higher $P\{\text{CS}\}$ when the simulation budget is increased. The orginal MOCBA and the simplified version always perform better than equal allocation in terms of the obtained $P\{\text{CS}\}$, or the total simulation budget in achieving a same level of probability of correct selection. It is also noted that in this case, MOCBA_simplifed performs slightly worse than

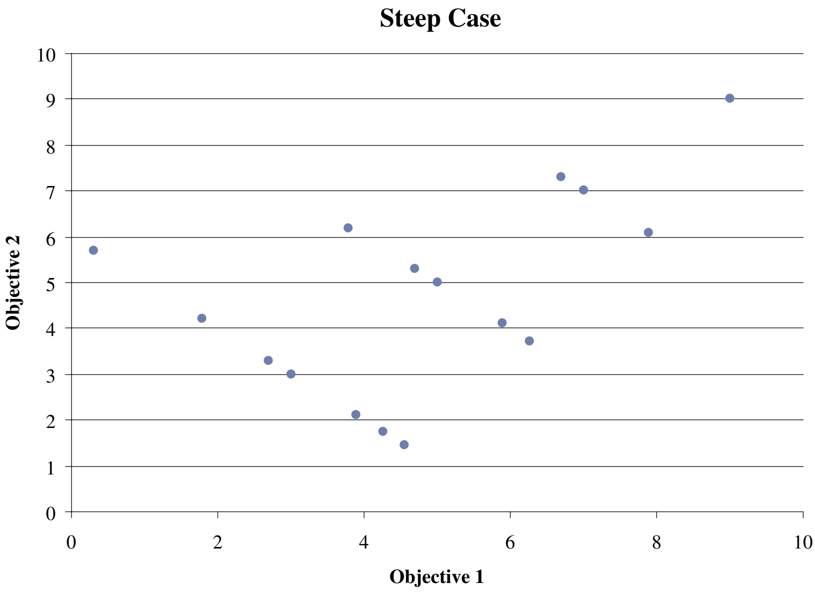


Figure 6.7. Designs with steep spread.

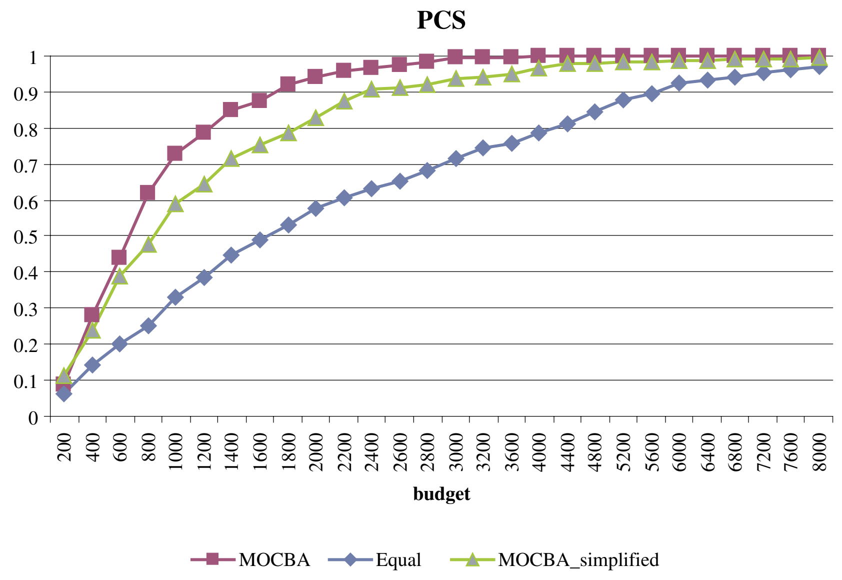


Figure 6.8. Comparison of $P\{\text{CS}\}$ for the steep case.

Table 6.6. Comparison of simulation budgets for the steep case.

$P\{\text{CS}\}$	MOCBA	simplified	Equal	Saving	Saving_simplified
0.90	1700	2400	5600	69.6%	57.1%
0.95	2100	3600	7200	70.8%	50.0%
0.97	2500	4200	8000	68.8%	47.5%

the original OCBA, which is attributed to the simplication or ignorance of some sampling information in the simulation runs. However, the simplified rule still performs significantly better than equal allocation, which implies that the simplified MOCBA can be easier to implement in practice, while not leading to significant compromise in performance.

Chapter 7

Large-Scale Simulation and Optimization

We extend the idea of efficient computing budget allocation to large-scale simulation optimization problems. The setting is the same general optimization problem defined in Chapter 1, which is to find a configuration or design that minimizes the objective function:

$$\min_{\theta \in \Theta} J(\theta), \tag{7.1}$$

where θ is a p-dimensional vector of all the decision variables, and Θ is the feasible region. What we meant by "large scale" is that Θ is discrete and large or continuous. In previous chapters, Θ is sufficiently small so that enumeration is possible. When Θ is large, the number of alternative designs is so high that simulating *all* designs becomes very expensive or even infeasible. Some sorts of sampling must be applied to explore the design space Θ.

For these problems, if one is willing to compromise on the optimality, ordinal optimization suggests that we need only to sample and simulate a relatively small number of designs in order to find a good one (cf. [Ho *et al.*, 1992; Dai and Chen, 1997; Luo *et al.*, 2001]). Instead of finding the best design, ordinal optimization concentrates on finding a good enough design within a significantly reduced computation time. Furthermore, ordinal optimization can give a measure of the quality of the best design found. We denote

$p \equiv \dfrac{|G|}{|\Theta|}$: the population proportion of good feasible designs,

where G is the population of good feasible designs. $|\Theta|$ is extremely large. Any design in G is a "good" design. Instead of finding the best design in Θ, the goal is to find a good design in G quickly. If we randomly sample a design, then let the probability that this sample is in G be p. If we randomly sample m designs:

$$\begin{aligned} &P\{\text{at least one of the selected designs is in } G\} \\ &\quad = 1 - P\{\text{none of the selected } m \text{ designs is in } G\} \\ &\quad = 1 - (1-p)^m. \end{aligned}$$

If we want to guarantee that at least one of the selected designs is in G with a probability at least as high as P_{sat}, then

$$P_{\text{sat}} = 1 - (1-p)^m. \tag{7.2}$$

Solving for m in Equation (7.2), one obtains

$$m = [\ln(1 - P_{\text{sat}})]/\ln(1-p). \tag{7.3}$$

Table 7.1 shows how to interpret Equation (7.3) for different values of P_{sat} and p. For example, if one wants the probability of a selected best design within the top 0.1% (99.9 percentile) be as high as 99%, i.e., $P_{\text{sat}} = 0.99$ and $p = 0.001$, one needs only $m = [\ln(1-0.99)]/\ln(1-0.001) = 4602.8 \approx 4603$ sampling designs. Note that for a given P_{sat}, the number of samples needed goes up roughly by a factor of 10 in order to reduce p by a factor of 10. Also note that this number is independent of the size of the design space $|\Theta|$.

The numbers in Table 7.1 are relatively small, and at any rate much smaller than the size of the design space for any reasonably sized combinatorial problem. Thus ordinal optimization allows us to quickly isolate with a high probability a good design, even with a blind sampling scheme. For further details about ordinal optimization, please refer to Ho *et al.* [2007], which offers an excellent and comprehensive coverage.

Table 7.1. Required number of designs for applying ordinal optimization.

P_{sat}	Top 1%	Top 0.1%	Top 0.01%
50%	69	692	6932
90%	229	2301	23025
99%	459	4603	26049
99.9%	688	6905	69075
99.99%	917	9206	92099
99.999%	1146	11508	115124

Table 7.1 provides a guide on the number of designs which we have to sample in order to guarantee a selection of a good enough design. However, with stochastic simulation noise, we have to simulate all the sampled designs with multiple replications in order to find the best from these sampled designs. For example, if one wants the probability of a selected best design within the top 0.1% to be as high as 99%, one needs to sample only 4603 designs. However, we still have to simulate these 4603 sampled designs and find the best one. This is actually a standard OCBA problem. The OCBA algorithms presented in previous chapters can be applied to enhance the simulation efficiency.

Furthermore, there is room for enhancement beyond uniform random (blind) sampling scheme. In many applications, knowledge of the problem might be used to orient the direction of sampling/search. The information obtained from earlier samples may also be used to guide the search. This can be viewed as biased sampling, which can be achieved by using expert knowledge to reduce the search space, and using metaheuristics framework to explore the design space in a more effective way. The goal is to obtain better results with the same number of sampled designs. In this case, the required number of samples given in Table 7.1 serve as a lower bound when one searches for a good design.

Note that in some problems, the gradient search method can be employed to search the design space when the performance of the designs can be described using a metamodel. However, in this chapter, our focus is on those problems where such a metamodel cannot be found easily.

7.1. A General Framework of Integration of OCBA with Metaheuristics

When the design space Θ is large, instead of simulating all possible designs, some optimization search methods can be applied to search the design space. By exploring the information in the design space, the expensive enumeration can be avoided. Further, some prescreening can be taken before the search starts so that the problem becomes simpler. We first illustrate the approach by considering a hypothetical simulation optimization example where there are 12 decision variables. Each decision variable has 10 alternative choices. In total, there are 10^{12} designs. It is certainly too expensive to simulate all the 10^{12} designs to find the best design.

Among these 12 decision variables, some of them are not critical or sensitive to the objective of the optimization problem. Those non-critical variables can be identified using some pre-screening techniques (cf. [Goldsman and Nelson, 1998; Wan *et al.*, 2003]) or design of experiments (cf. [Berger and Maurer, 2002; Sanchez, 2005]). Suppose 3 of the 12 decision variables are determined to be non-critical variables and so can be excluded from further optimization process. By focusing on the remaining 9 important variables, we have 10^9 designs. While 10^9 is much smaller than 10^{12}, it is still too expensive to simulate all the 10^9 designs in order to find the best design. Some sort of optimization search algorithms must be applied in order to iteratively search the design space to find the optimum as depicted in Figure 7.1. At each iteration of the search, several designs are selected for simulation. The simulation output information is used to guide the search for next iteration. Such simulation and search iteratively proceeds until a good or optimal design is found.

Such a framework involves two major issues:

- How should we iteratively search the design space Θ? Metaheuristics aim to address this issue.

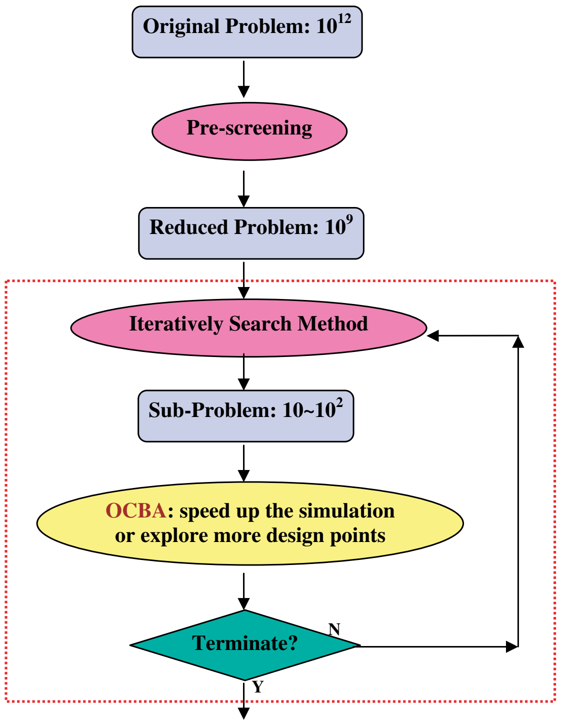

Original Problem: 10^{12}
Pre-screening
Reduced Problem: 10^9
Iteratively Search Method
Sub-Problem: $10\sim10^2$
OCBA: speed up the simulation or explore more design points
Terminate?
N
Y

Figure 7.1. An illustrative example of integrating OCBA with search methods for large-scale simulation optimization problems. The box area is further explained in Figure 7.2.

- How should we efficiently simulate those designs generated in each iteration in order to provide the necessary information to guide the new search? Ideally, we want to simulate the generated designs in an intelligent way so that the total simulation cost is minimized.

This book focuses more on the second issue. We would like to find an intelligent way to allocate the simulation budget so that the necessary simulation output can be obtained efficiently to guide the search effectively given a metaheuristic. This budget allocation scheme should consider the information needed in the metaheuristic. OCBA can serve this purpose. The integration framework of OCBA

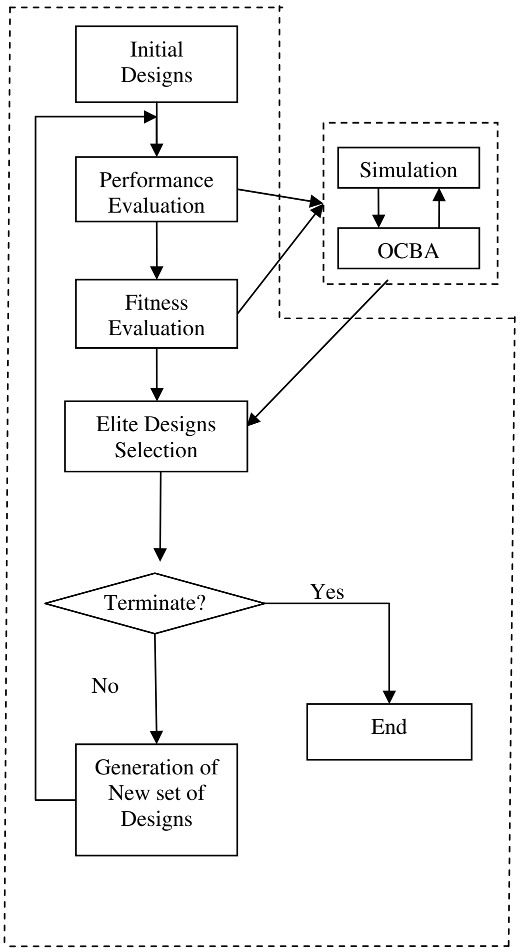


Figure 7.2. An example illustrating how OCBA helps a search method for large-scale simulation optimization problems.

and metaheuristics is shown in Figure 7.2. Most of the metaheuristics start with a population of designs, and elite designs will be selected from this population of designs in order to generate the better population during the search process. Evaluation via simulation is critical in determining the elite designs, and OCBA is able to help to conduct the evaluation effectively by allocating computing budget intelligently. In the remaining sections of this chapter, we will provide examples of how OCBA can be integrated with the existing metaheuristics. Section 7.2 provides examples of problems with a single objective. The numerical testing in Section 7.3 demonstrates that the efficiency can be enhanced using this framework. Section 7.4 presents integration examples for problems with multiple objectives.

7.2. Problems with Single Objective

In this section, we will give a brief introduction to several metaheuristics or search methods and show how OCBA procedures can be easily integrated with them to enhance efficiency. The metaheuristics considered in this chapter include

- Neighborhood Random Search (Section 7.2.1),
- Cross-Entropy Method (Section 7.2.2),
- Population-based Incremental Learning (Section 7.2.3), and
- Nested Partitions (Section 7.2.4).

As we will show, the integration with OCBA is straightforward and effective. It is worth noting that the applicability of OCBA is not limited to the methods presented here. The same idea is applicable to many other metaheuristics or search methods.

7.2.1. *Neighborhood random search* (*NRS*)

We start with a simple random search method. This method can be implemented very easily. The primary assumption is that the neighborhood of an elite design has a higher chance of including the optimal design than other neighborhoods. Therefore we want to spend more efforts searching in neighborhoods of elite designs, and less efforts on neighborhoods of non-elite designs.

Specifically, in each iteration, k alternative designs are simulated and then the top-m designs are selected ($m < k$). For the next iteration, a large proportion of candidate designs is generated by sampling the neighborhood of the elite designs. The longer the distance, the smaller the probability a design is sampled. The remaining (smaller) portion of samples are taken from the entire design space to ensure convergence to the global optimum. The algorithm is summarized as follows.

Step 1. **Initialization.** Uniformly sample k candidate designs over the design variable space.
Step 2. **Termination.** Stop, if the stopping criterion is met.
Step 3. **Evaluation and Selection.** Simulate the sampled k alternatives and select a subset containing the top-m.
Step 4. **Generation of New Population.** Sample 80% of the new k candidate designs in the neighborhood of the top-m elite designs. Another 20% are taken uniformly from the entire design space. Go back to Step 2.

In this algorithm, Step 3 is the most critical because it provides information to guide the search for the next iteration. It is also the most time consuming step because simulation for the k alternative designs must be performed in order to correctly select the elite subset of the top-m designs. If the simulation cost is not cheap, the computation cost for other steps is negligible, compared with that for Step 3. Then the overall efficiency depends on how efficiently we simulate the candidate designs to correctly select the elite set. Our goal herein is to allocate the computing budget in the most efficient manner so that the elite top-m can be identified. The OCBA-m algorithm presented in Chapter 5 works perfectly here.

7.2.2. *Cross-entropy method (CE)*

The Cross-entropy (CE) method (cf. [Rubinstein and Kroese, 2004]) was originally used for finding the optimal importance sampling measure in the context of estimating rare event probabilities. Later it was developed into a technique for solving optimization problems. It works with a parametrized probability distribution. In every iteration of CE, we will first generate a population of designs from a probability density function (pdf) with a certain parameter. After all the designs in this population have been evaluated (via simulation in our case), we will select the elite designs. These elite designs will then be used to update the parameters of this pdf, which will be used to generate the population for the next iteration.

In updating the parameters, we need to solve an optimization model which minimizes the Kullback–Leibler divergence (or the cross entropy) between the unknown optimal sampling distribution and the parametrized distribution. This optimization model depends only on the designs in the elite subset. Hence to update the parameters, we need to identify the top-m design efficiently and correctly for each newly generated population.

The algorithm is summarized as follows.

Step 1. **Initialization.** Initialize a sampling distribution with parameter P.
Step 2. **Generation of New Population.** Sample k candidate designs using the sampling distribution with parameter P.
Step 3. **Evaluation and Selection.** Simulate these sampled k alternative designs and select a subset containing the top-m.
Step 4. **Parameter Updating.** Update P based on the selected top-m by solving the optimization model which minimizes the Kullback–Leibler divergence.
Step 5. **Termination.** Go back to Step 2 if the stopping criterion is not met.

Like the Neighborhood Random Search method, in this algorithm, Step 3 is the most critical and time consuming step in which k designs must be simulated to correctly identify the elite subset of the top-m designs. The OCBA-m algorithm presented in Chapter 5 can be applied here.

7.2.3. *Population-based incremental learning (PBIL)*

The PBIL algorithm was developed for binary search problems [Baluja, 1994] and continuous optimization problems [Rudlof and Köppen, 1996]. The PBIL updates the sampling distribution P with a probabilistic learning technique using the estimated mean of an "elite" subset of good candidate designs in each iteration. This algorithm is almost the same as the CE algorithm, except in Step 4, the PBIL learning principle is applied. The algorithm is summarized as follows.

Step 1. **Initialization.** Initialize a sampling distribution P.
Step 2. **Generation of New Population.** Sample k candidate designs using P.
Step 3. **Evaluation and Selection.** Simulate these sampled k alternative designs and select a subset containing the top-m.
Step 4. **Parameter Updating.** Update P based on the selected top-m and the PBIL principle.
Step 5. **Termination.** Go back to Step 2 if the stopping criterion is not met.

Similar to NRS and CE methods, in Step 3 of this algorithm, k designs must be simulated so that the elite subset of the top-m designs can be identified correctly, which takes most of the computational effort. The OCBA-m algorithm presented in Chapter 5 is directly applied here to enhance efficiency.

7.2.4. *Nested partitions*

The Nested Partitions (NP) method has recently been proposed to solve global optimization problems (see [Shi and Olafsson, 2000, 2008; Fu *et al.*, 2008] for more details). The method can be briefly described as follows. In each iteration, a region considered most promising is assumed. We then partition this region into M subregions and aggregate the entire surrounding region into one. Therefore, within each iteration, we only look at $M+1$ disjoint subsets that cover the feasible space. Each of these $M+1$ regions is sampled using some random sampling scheme and the estimated performance function values at randomly selected design points are used to approximate a so-called promising index for each region. This index determines which region becomes the most promising one in the next iteration. The sub-region scoring highest on the promising index becomes the most promising region in the next iteration. The new most promising region is thus nested within the last. The method backtracks to a larger region, if the surrounding region rather than a sub-region is found to have the best promising index. The new most promising region is then partitioned and sampled in a similar fashion. The partitioning continues until singleton regions are obtained and no further partitioning is possible. The partitioning strategy imposes a structure on the feasible region and is therefore important for the rate of convergence of the algorithm. If the partitioning is such that most of the good solutions tend to be clustered together in the same subregions, it is likely that the algorithm quickly concentrates the search in these subsets of the feasible region.

One common definition of promising index is the performance measure of the best sampled design in that subregion. Thus, the most promising region is the one containing the best sampled design among all the subregions. With this choice of a common promising index, determination of the most promising region is equivalent to the determination of the best sampled designs. The algorithm is summarized as follows.

Step 1. **Initialization.** Let Θ be the promising region.

Step 2. **Partition.** Partition the current most promising region into M sub-regions and aggregate the surrounding region into one.

Step 3. **Generation of New Population.** Randomly sample q designs from each of the subregions and from the aggregated surrounding region; $k = (M + 1)q$.

Step 4. **Evaluation and Selection of the Most Promising Region**. Simulate these k sampled designs and select the best among the k designs. The subregion containing this best design becomes the most promising region.

Step 5. **Termination.** Stop, if the stopping criterion is met.

Step 6. **Determination of Further Partition or Backtracking.** If one of the M subregions becomes the most promising region, the algorithm moves to this region for further partitioning in the next iteration. If the surrounding region contains the best design, the algorithm backtracks to a larger region. If the new promising region contains more than one design, go back to Step 2.

Step 4 is to simulate and select the best from the set of sampled designs, comprising the union of all sampled designs from each disjoint (sub)region. It is essentially a small and discrete simulation optimization problem discussed in earlier chapters. This step involving many replications of simulation usually consumes most computational effort. Our goal herein is to allocate the computing budget in a most efficient manner so that the best design can be identified. Hence, the OCBA method presented in Chapter 3 can be applied directly. Details about this integration of OCBA and NP can also be found in Shi and Chen [2000].

7.3. Numerical Experiments

To illustrate improvements in efficiency, we carried out numerical experiments for two typical optimization problems using some procedures presented in this chapter. Most of the numerical setting is the same as those presented in Chapters 4 and 5. Further details can also be found in Chen *et al.* [2008]. Three search methods are tested:

- Neighborhood Random Search (NRS),
- Cross-Entropy Method (CE),
- Population-based Incremental Learning (PBIL).

The OCBA-m procedure is integrated into each of the three search algorithms, and the resulting performance of the algorithm is compared with the same algorithm without using OCBA, in which case we apply equal simulation of all searched candidate designs.

Experiment 7.1. Griewank function

The Griewank function is a common example in the global optimization literature (cf. [Fu *et al.* 2006]), given in two-dimensional (2-D) form by

$$f(x_1, x_2) = \frac{1}{40}\left(x_1^2 + x_2^2\right) - cos(x_1)cos\left(\frac{x_2}{\sqrt{2}}\right) + 1,$$

where x_1 and x_2 are continuous and $-10 \le x_1 \le 10$, $-10 \le x_2 \le 10$. The unique global minimum of this function is at $(x_1^*, x_2^*) = (0, 0)$ and $f(x_1^*, x_2^*) = 0$. The additive noise incurred in stochastic simulation

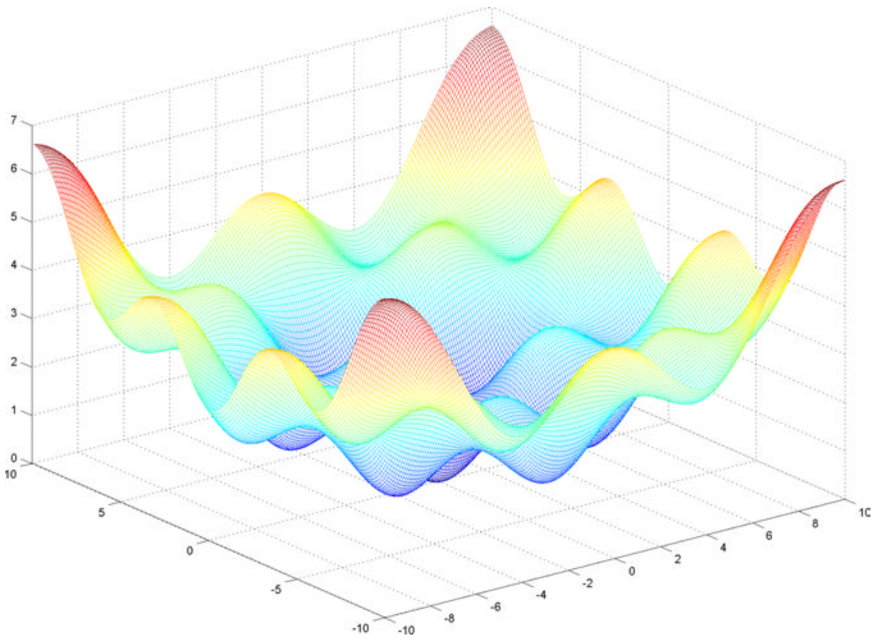


Figure 7.3. 2-D Griewank function tested in Experiment 7.1.

is $N(0,\ 1^2)$. Figure 7.3 gives an illustration of this function without simulation noise.

In numerical implementation, the stopping criterion for the evaluation-and-selection problem in Step 3 is when the posterior $APCSm$ given by Lemma 3.2 is no less than $1 - 0.2^{*}\exp(-q/50)$, where q is the iteration number. We set $k = 100$ and $m = 5$. In comparing the procedures, the measurement of effectiveness used is the average error between the best design thus far and the true optimal design over 200 independent experiments. The results are shown in Figure 7.4. The thick lines indicate the performance with OCBA-m for different optimization algorithms, while the thin lines show the performances without OCBA-m. Lines with different patterns represent the use of different optimization search algorithms.

We see that the optimality gap decreases for all procedures as the available computing budget increases. In this example, NRS performs better than PBIL, which does better than CE. However, OCBA-m significantly enhances the efficiency for all three search methods. For

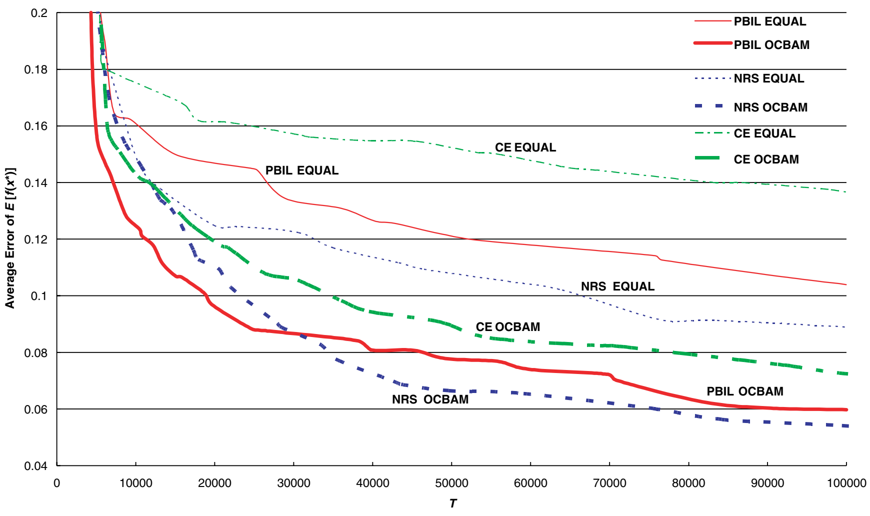


Figure 7.4. Performance comparison for three optimization search algorithms with and without OCBA-m for Experiment 7.1.

example, with integration of OCBA-m, NRS can achieve an average error of 0.1 using a computation cost of 22,800. Without OCBA-m, NRS spends a computation cost of 68,500 to achieve the same level of error. Similarly, the computation costs for PBIL to reduce the average error to 0.12 with and without OCBA-m are 11,500 and 53,700, respectively. The speedup factor of using OCBA-m is even larger if the target level of optimality is higher.

Experiment 7.2. Rosenbrock function

The Rosenbrock function is another common example in the global optimization literature (e.g., [Fu *et al.*, 2006]). It is a non-convex function with a "banana-shaped" valley given in 2-D by

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2,$$

where x_1 and x_2 are continuous and $-5 \leq x_1 \leq 5$, $-5 \leq x_2 \leq 5$. The global minimum of this function is at $(x_1^*, x_2^*) = (1, 1)$ and $f(x_1^*, x_2^*) = 0$. The additive noise incured in stochastic simulation is $N(0, 10^2)$. Figure 7.5 gives an illustration of this function without noise.

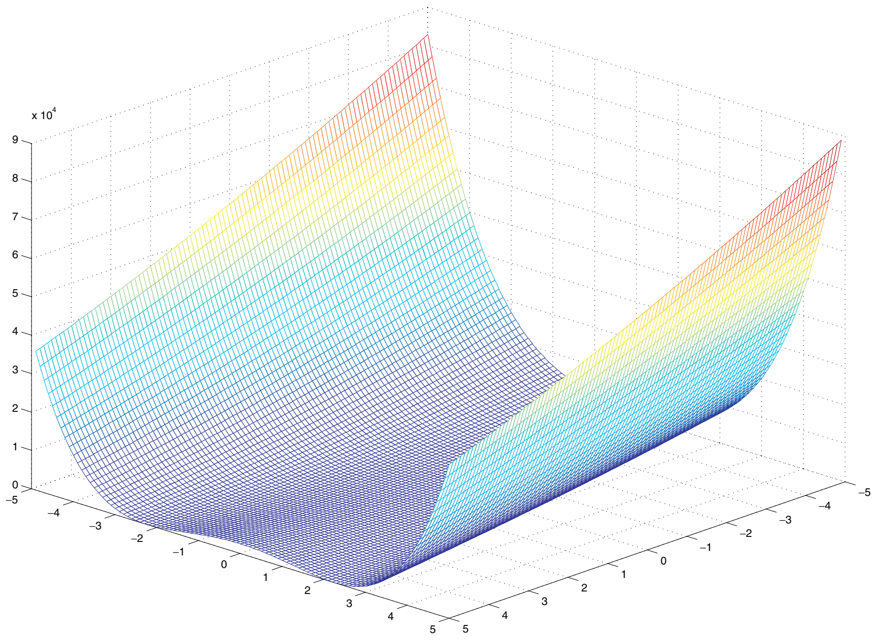


Figure 7.5. 2-D Rosenbrock function tested in Experiment 7.2.

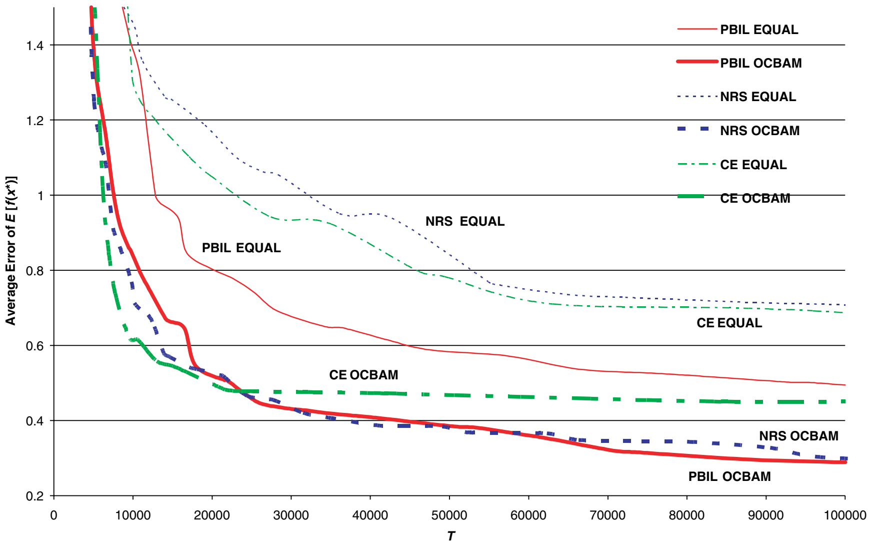


Figure 7.6. Performance comparison for three optimization search algorithms with and without OCBA-m for Experiment 7.2.

The numerical setting is the same as that in Experiment 7.1. The test results are shown in Figure 7.6. Unlike Experiment 7.1, the PBIL method has the best performance in this example. Although the order of optimization methods are different from that in Experiment 7.1, the level of efficiency enhancement using OCBA-m is very similar.

While different optimization algorithms perform differently in the preliminary numerical examples, the OCBA allocation significantly enhances the computational efficiency for each individual search algorithm.

7.4. Multiple Objectives

Section 7.2 deals with problems with a single objective function to optimize. In this section, we turn our attention to cases where we are facing multiple objectives, and we would like to find Pareto optimal solutions. The search method designed for the single-objective problem can be directly employed in this case except that we have to redefine the fitness function. In Chapter 6, we have defined the probability of non-dominating for each design as ψ, which can be used as the fitness function to rank the designs for multiple-objective problem. However, as there is no closed-form expression for ψ, we will use its upper bound to approximate this fitness function.

In the next two sub-sections, we will show how to integrate MOCBA presented in Chapter 6 with the metaheuristics. The metaheuristics considered here are

- Nested Partitions (Section 7.4.1), and
- Evolutionary Algorithm (Section 7.4.2).

7.4.1. *Nested partitions*

The method of NP has been introduced in Section 7.3.4. Two changes in the integrated NP procedure must be made. First, the performance measure for each design is changed to the probability of non-dominating, and so, like the single-objective case, the promising index of a subregion becomes the highest probability of non-dominating among all the designs in that subregion. Second, we need to keep an elite set which consists of all Pareto optimal solutions.

The algorithm is summarized as follows.

Step 1. **Initialization.** Let Θ be the promising region, and Ω be the elite set.

Step 2. **Partition.** Partition the current most promising region into M sub-regions and aggregate the surrounding region into one.

Step 3. **Generation of New Population.** Randomly sample q designs from each of the subregions and from the aggregated surrounding region; $k = (M + 1)q$.

Step 4. **Evaluation and Selection of the Most Promising Region.** Simulate these k sampled designs and select the Pareto optimal designs from them. Compare the new Pareto optimal designs with the designs in the elite set obtained in the previous iteration, and update the elite set so that it only consists of updated Pareto optimal solutions thus far. The subregion containing the best sampled design (the one with the highest probability of non-dominating) becomes the most promising region.

Step 5. **Termination.** Stop, if the stopping criterion is met.

Step 6. **Determination of Further Partition or Backtracking.** If one of the M subregions becomes the most promising region, the algorithm moves to this region for further partitioning in the next iteration. If the surrounding region contains the best design, the algorithm backtracks to a larger region. If the new promising region contains more than one design, go back to Step 2.

Step 4 of the algorithm is to simulate all the designs in order to identify the Pareto optimal designs. This step consumes most computational effort and the MOCBA method presented in Chapter 6 can be applied directly here to improve the simulation efficiency. Details about this integration of MOCBA and NP can also be found in Chew *et al.* [2009].

7.4.2. *Evolutionary algorithm*

Evolutionary Algorithm (EA) is an adaptive heuristic search algorithm which simulates the survival of the fittest among individuals over consecutive generations for optimization problem solving. Based on an initial population randomly generated, at each generation, EA evaluates the chromosomes and ranks them in terms of their fitness; the fitter solutions will be selected to generate new offspring by recombination and mutation operators. This process of evolution is repeated until the algorithm converges to a population which covers the non-dominated solutions. EA has been successfully applied in solving multi-objective problems [Fonseca and Fleming, 1995; Hanne and Nickel, 2005].

To make EA work well for simulation-based problems where stochastic noise is a main concern, MOCBA is needed mainly in the following two aspects: fitness evaluation and elite population formation. The integration algorithm is summarized as follows.

Step 1. **Initialization.** Randomly sample k designs from the design space Θ to form an initial population.

Step 2. **Evaluation and Selection of the Pareto Optimal Designs**. Simulate these k sampled designs and select the Pareto optimal designs from them. Compare these Pareto optimal designs obtained in this iteration with the designs in the earlier elite set, and update the elite set so that it only consists of updated Pareto optimal solutions thus far.

Step 3. **Elite Set Updating.** Compare the new Pareto optimal designs obtained in this iteration with the designs in the earlier elite set, and update the elite set so that it only consists of updated Pareto optimal solutions thus far.

Step 4. **Termination.** Stop, if the stopping criterion is met.

Step 5. **Generation of New Population by Crossover and Mutation Operation.** Randomly select two candidate designs from the elite set to perform the crossover operation. Repeat this until there are enough candidate designs to form the new population. Randomly select several candidates from this new population, and perform mutation operations. Go back to Step 2.

Step 2 of the algorithm involves many replications of simulation for all designs in the new population and so consumes most of the computational effort. The MOCBA method presented in Chapter 6 can be applied directly here to improve simulation efficiency. Details about this integration of MOCBA and EA can also be found in Lee *et al.* [2008].

7.5. Concluding Remarks

As shown in all the previous sections, OCBA, OCBA-m, and MOCBA can be applied directly in the search methods. This is because these search methods need to find the best, top-m or Pareto optimal designs in order to determine the new search direction for the next iteration. However, some search methods may require more information when determining the new search direction, in which case the existing OCBA algorithms might not be sufficient, and so newer OCBA algorithms must be developed according to what is needed in the specific search method. In Section 8.2, we will provide such an example to illustrate how a new OCBA algorithm is developed. This example utilizes an extended cross-entropy method.

When the search method is chosen and its stopping criterion of the evaluation in each iteration is determined, the quality of the selected designs used to guide the new search is pretty much set. It implies that the efficacy of the search is fixed. The benefit of OCBA in this integration is the savings of computing time in evaluating the sampled designs. This saving in time can be utilized in a few ways:

- Terminating the search earlier,
- Using the time saved to generate more sampled designs in an iteration, or
- Running the search algorithm with more iterations.

Either of these should improve the search quality or efficiency, resulting in a better design found eventually.

In this chapter, the simulation optimization problem can be decomposed into a search part and evaluation part because the stopping criterion of the evaluation part is fixed. OCBA is applied to the evaluation part to ensure the selection quality is reached in a minimum time. However, there exists a trade-off between the selection quality and search quality given a fixed total computing budget. Overall, there are three issues.

1. How many iterations should the search algorithm run?
2. How many designs should be sampled in each iteration?
3. How much budget should be allocated to evaluate the sampled designs?

Ideally we want to allocate the computing budget in an optimal way by considering all three issues together. Due to the complex nature of this problem, it remains a research topic.

Chapter 8

Generalized OCBA Framework and Other Related Methods

The previous chapters focus on development of OCBA algorithms for stochastic simulation optimization problems, particularly for selection of the best design, an optimal subset of designs, or Pareto optimal designs. In fact, the notion and mathematical framework of "Optimal Computing Budget Allocation" can also be applied to different problems or domains beyond stochastic simulation optimization.

A generalized view of the OCBA framework is shown in Figure 8.1. The total budget T is allocated to all the different processors (represented by rectangular boxes), where each processor will be assigned with a budget N_i. Each processor generates an output X_i whose quality depends on the budget allocated, i.e., N_i. Usually the higher the budget allocated, the better the quality of the output. Then these outputs will be processed by a synthesizer which produces an overall outcome. With a fixed total computing budget, different budget allocation will result in different quality of the outputs and hence the outcome. The generalized OCBA model is an optimization model which determines the best allocation scheme to maximize a certain objective which is related to the quality of the outcome.

In addition, the figure on the cover of this book offers another intuitive illustration of the OCBA question. We want to optimally

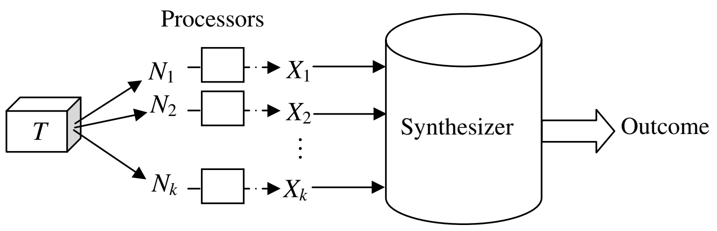


Figure 8.1. A generic view of the OCBA framework.

determine which processor should receive each portion of the computing budget.

For the case of simulation optimization, T is the total number of simulation replications which can be run, the processor is a simulation engine, the output from the processor is the simulation output, the synthesizer is a selection procedure which uses the simulation outputs to select the best design, and the outcome is the selected design which is hopefully the true best design. The allocation quantity N_i (i.e., the number of simulation replications) has impact on the variability of the simulation output (X_i), which has subsequently impact on the quality of the selected design, usually measured by a probability of correct selection ($P\{\text{CS}\}$). The overall goal of OCBA is to determine the optimal simulation allocation quantity in order to achieve the best outcome, $P\{\text{CS}\}$. Similarly, we can also view OCBA-m and MOCBA using this generalized framework. All elements are the same except that the synthesizer tries to identify the top-m or Pareto optimal designs. Since the desired outcomes are different, naturally, the optimal simulation allocations are different as shown in previous chapters.

Based on this generalized view of OCBA, we can apply this framework beyond simulation optimization problems discussed in the previous chapters. The question is how we should allocate a general budget so as to achieve the best outcome for a specific problem. The first problem shown in Section 8.1 illustrates how we can extend the OCBA notion to problems where the performance of the simulation output follows some functional relationship, and regression can be used to estimate this relationship. In this case all elements in the framework are the same except the synthesizer needs to perform a regression analysis before making the selection.

In the second problem, we utilize the framework to develop a new OCBA algorithm for the extended Cross Entropy (CE) method, which requires the estimation of the coefficients to minimize the Kullback–Leibler divergence. To view this problem using the generalized framework, it is similar to the OCBA-m except that in the outcome, we need to not only identify the top-m designs, but also accurately estimate their performances. The objective of the synthesizer is to minimize the mean squared error for the coefficients, which rely on the correct estimate of the performance of the top-m designs. As shown in Section 8.2, the budget allocation rule obtained will be different from OCBA-m due to the different requirement.

In Section 8.3, we extend the notion of OCBA to the problem of estimating a rare-event probability using simulation. This is not an optimization problem. The splitting method is a popular variance reduction technique which splits the original simulation into several levels, each of which is smaller and easier to simulate than the original. In this case, N_i is the number of simulation replications for level i, the processor is actually the simulation of one level, X_i is the estimated conditional probability estimated using simulation at level i and the synthesizer is to estimate the overall rare-event probability. The OCBA is to determine the optimal budget allocated to each level of simulation in order to minimize the variance of the rare-event probability estimation.

The OCBA concept and framework can also be extended to problems without simulations or optimization. Section 8.4 shows an example on how OCBA can help determine the budget allocation for data collection which maximizes the accuracy of an efficiency prediction in data envelopment analysis (DEA). By viewing this problem using the generalized OCBA framework, T is the total budget for data collection, N_i is the number of data points allocated for unknown variable i, processor represents the actual data collection, X_i is the sample average of the data collected, synthesizer is a Monte Carlo DEA model whose outcome is the predicted efficiency, and the OCBA objective is to minimize the mean squared error of the prediction.

Table 8.1 summarizes variant OCBA problems discussed in the book using this generalized framework. Note that some of these problems are related to simulation and/or optimization; but some are not. These examples are mainly for the purpose of illustration; the generalization is not limited to these examples. Readers can certainly extend the OCBA notion and framework to many other problems with some development.

8.1. Optimal Computing Budget Allocation for Selecting the Best by Utilizing Regression Analysis (OCBA-OSD)

Similar to the problem addressed in Chapter 3, in this problem, the goal is also to identify the best design among k alternatives except that an underlining quadratic function is assumed for the performance of these designs.

Let x_i be the location for design i, which is a real number, $y(x_i)$ be the performance for design i, and b be the design with the best predicted performance. The OCBA model, also known as OSD method (Optimal Simulation Design), is as follows.

$$\begin{aligned} \max_{N_1,\ldots,N_k} \quad & P\{\mathrm{CS}\} = P\{y(x_b) \le y(x_i)\, \forall\, i\ \}, \\ \text{s.t.} \quad & N_1 + N_2 + \cdots + N_k = T. \end{aligned} \tag{8.1}$$

The model defined in Equation (8.1) is actually the same OCBA problem defined in Chapter 3 except that a regression is performed to estimate $y(x_i)$ before the selection of design b takes place. Due to the addition of regression analysis, the formulation for $P\{\mathrm{CS}\}$ is different and so the solution will be different.

By exploiting the structure of the quadratic function, Brantley *et al.* [2008] show that the required number of comparisons in the $P\{\mathrm{CS}\}$ equations expressed in Equation (8.1) can be reduced from the $k-1$ comparisons to 2 comparisons (comparisons with design $b-1$ and $b+1$), if b is not at a boundary location. Moreover, they show

Table 8.1. Summary of OCBA problems.

Type	T	N_i	Processor	X_i	Synthesizer	Outcome	Objective
OCBA	Computing budget	simulation replication number	simulation	simulation output (mean)	selecting the best	selected best	PCS(best)
OCBA-m	Computing budget	simulation replication number	simulation	simulation output (mean)	selecting top-m	selected top-m	PCS(top-m)
MOCBA	Computing budget	simulation replication number	simulation	simulation outputs (mean)	selecting Pareto Optimal	selected Pareto Optimal	PCS(Pareto Set)
OCBA-CO	Computing budget	simulation replication number	simulation	simulation outputs (mean)	selecting feasible best	selected feasible best	PCS(feasible best)
OCBA-OSD	Computing budget	simulation replication number	simulation	simulation output (mean)	selecting the best after regression	selected best	PCS(best)
OCBA-CE	Computing budget	simulation replication number	simulation	simulation output (mean)	selecting top-m	Selected top-m, and their performance	Mean squared error (coefficient of KL divergence)
OSTRE	Computing budget	simulation replication number for partial simulation	partial simulation	conditional probability	Estimating rare-event probability	Estimated rare-event probability	Var(estimated rare-event probability)
ODCBA	Data Collection Budget	Number of data points for each unknown variable	data collection	sample average	Monte Carlo DEA to estimate the efficiency	Estimated Efficiency	Mean Squared Error (efficiency)

that there are at most 3 design locations which should receive some of the simulation budget. These three designs, also called support points, are determined as follows.

Two of the three support points are always the first and the last point, i.e., x_1 and x_k, respectively. The third support point is

$$x_s = \begin{cases} x_{\mathrm{M}+b-1}, & \dfrac{3x_1 + x_k}{4} \leq \dfrac{x_{\mathrm{M}} + x_b}{2} < \dfrac{x_1 + x_k}{2} \\ x_{\mathrm{M}+b-k}, & \dfrac{x_1 + x_k}{2} < \dfrac{x_{\mathrm{M}} + x_b}{2} \leq \dfrac{x_1 + 3x_k}{4} \\ x_{(k-1)/2}, & \text{otherwise} \end{cases}$$

The optimal computing budget allocation for these three support points (i.e., N_1, N_s, and N_k) is determined as follows.

$$N_i = \frac{|D_{\mathrm{M},i}|}{|D_{\mathrm{M},1}| + |D_{\mathrm{M},s}| + |D_{\mathrm{M},k}|},$$

where

$$\mathrm{M} = \underset{i=b-1,b+1}{\arg\min} \left\{ \frac{\hat{d}(x_i)}{\sqrt{\varsigma_i}} \right\},$$

$$D_{i,k} = \left\{ \frac{(x_1 - x_i)(x_s - x_i) - (x_1 - x_b)(x_s - x_b)}{(x_k - x_1)(x_k - x_s)} \right\},$$

$$\varsigma_i = \sigma^2 \left[\frac{D_{i,1}^2}{N_1} + \frac{D_{i,s}^2}{N_s} + \frac{D_{i,k}^2}{N_k} \right],$$

$$\hat{d}(x_i) \equiv \hat{y}(x_i) - \hat{y}(x_b),$$

and $\hat{y}(.)$ is the predicted performance using regression.

The details of the work and the optimal allocation for the case when b is at the boundary location can be found in Brantley *et al.* [2008], which also show that the numerical performance can be dramatically enhanced due to the use of regression. Further, they also show that OSD outperforms traditional regression techniques (such as D-Optimal in design of experiments) because of the optimal computing budget allocation.

8.2. Optimal Computing Budget Allocation for Extended Cross-Entropy Method (OCBA-CE)

As discussed in Section 7.2.2, when updating the parameters for CE method, an optimization model which minimizes the Kullback–Leibler (KL) divergence needs to be solved. The model is given in Equation (8.2).

$$\nu_{t+1} = \arg\max_{v} \sum_{i=1}^{k_t} I_{i\in\Omega_t} \ln p(X_i, \nu), \tag{8.2}$$

where t is the iteration number, v_{t+1} is the updated parameter at iteration $t+1$, k_t is the number of designs generated at iteration t, I is the indicator function, Ω_t is the elite set which consists of the top-m designs found in iteration t, and $p(X_i, v)$ is the pdf of generating design X_i when the parameter of the distribution is v. Equation (8.2) can be interpreted as using the top-m designs to fit the sample distribution for the next iteration by applying a maximum likelihood estimation. Hence, it is critical for the CE method to correctly identify what the top-m designs are in order to correctly update the parameter v.

In some variants of CE methods, e.g., the extended CE method, a weight function is added in the optimization model as shown in Equation (8.3) to improve the convergence.

$$\nu_{t+1} = \arg\max_{v} \sum_{i=1}^{k_t} w(y_i) I_{\{i\in\Omega_t\}} \ln p(X_i, v), \tag{8.3}$$

where $w(y_i)$ is the weight function which depends on the performance of the design at X_i, denoted as y_i. One common choice for the weight function is the exponential function.

Due to this extra weight function, we need to not only correctly select the top-m designs but also accurately estimate their performances. OCBA-m is not sufficient for this purpose. To solve this problem, a new OCBA model is developed to determine the optimal simulation allocation. The framework of this new OCBA model is similar to the previous model except that the objective is no longer

$P\{\text{CS}\}$, instead it is the expected mean squared error which measures the accuracy of the coefficients in the extended CE updating function given in Equation (8.3). The OCBA objective is to optimally allocate the simulation budget so that the accuracy of these coefficients is maximized. The new OCBA model (OCBA-CE) is as follows.

$$\begin{aligned} \min_{N_1,\ldots,N_k} \quad & E\left[\frac{1}{k}\sum_{i=1}^{k}(w(\bar{X}_i)I_{\{i\in\Omega\}} - w(y_i)I_{\{i\in\Omega\}})^2\right], \\ \text{s.t.} \quad & \sum_{i=1}^{k} N_i = T. \end{aligned} \tag{8.4}$$

He *et al.* [2010] give an asymptotically optimal solution to this computing budget allocation problem when the weight function is an exponential function as follows.

- $\frac{N_i}{N_j} = \frac{e^{-ry_i}\sigma_i}{e^{-ry_j}\sigma_j}$, $i, j \in \Omega$,
- $\frac{N_i}{N_j} = \left(\frac{\sigma_i(\gamma-y_j)}{\sigma_j(\gamma-y_i)}\right)^2$, $i, j \notin \Omega$,
- $\frac{\sum_{i\in\Omega} N_i}{\sum_{j\in\Omega} N_j} \sim O\left(\frac{e^{\beta T}}{T^2}\right)$,

where β is a positive constant.

This allocation rule is similar to the OCBA-m rule for those designs which are not in the top-m set. However, the allocation ratio for those designs belonging to the top-m set depends on the weight function chosen. In general, the total budget allocated to the top-m designs will be asymptotically exponentially higher than the total budget allocated to other designs. This means when the computing budget is very large, almost all of the effort will be spent on the top-m designs. This is not hard to explain from ordinal optimization [Ho *et al.*, 1992], according to which the probability of correctly identifying the ranking of designs converges to 1 exponentially fast while the estimation accuracy converges only at the rate of $1/\sqrt{n}$, where n is the sample size. Since ranking converges much faster and we need both ranking and estimation accuracy, more effort must be devoted to enhancing the estimation accuracy of the top-m designs. Details can be found at He *et al.* [2010].

8.3. Optimal Computing Budget Allocation for Variance Reduction in Rare-event Simulation

There are many practical problems that involve the evaluation of rare-event probabilities. While simulation is a powerful tool that can be used to analyze a wide variety of systems, however, in the context of rare events, a key limitation is the computer time needed to obtain a reasonable estimate of the rare-event probability. For example, consider a rare event that occurs with probability 10^{-9}. If we simulate the system 10^9 times, then we will see, on average, one occurrence of this rare event. Even if we can simulate 10,000 runs per second, we need about 1 day of computer time just to observe one event. Furthermore, to obtain a reasonably tight confidence interval, many more simulations are needed. To address the computational issue, several variance reduction techniques have been developed, such as importance sampling (cf. [Glynn, 1994; Heidelberger, 1993]) and splitting (cf. [Glasserman *et al.*, 1999; Garvels *et al.*, 2002]). We will focus on the splitting technique here.

The basic idea of level splitting is to "split" a simulation into separate independent runs whenever it gets "near" the rare event of interest. Effectively, this multiplies runs that are promising to reach the rare-event and kills runs that are not promising, thus improving the likelihood of observing the rare-event. Figure 8.2 illustrates the idea. In the figure, the y-axis measures the proximity of the system to the rare event set. The process is assumed to start at state 0 and the rare event set is defined by a threshold level L. For example, the y-axis could denote the length of a queue and L could denote the maximum queue size before buffer overflow. The interval $[0, L]$ is partitioned into m stages by choosing levels $0 = L_0 < L_1 < \cdots < L_m = L$. Whenever the simulation crosses a level, it is "split" into separate simulation runs. These runs are independently simulated starting from the splitting point. In this way, more computer time is spent on runs which are closer to the rare event.

Let γ be the rare-event probability which is defined as the probability that the process enters the rare event set before returning to the

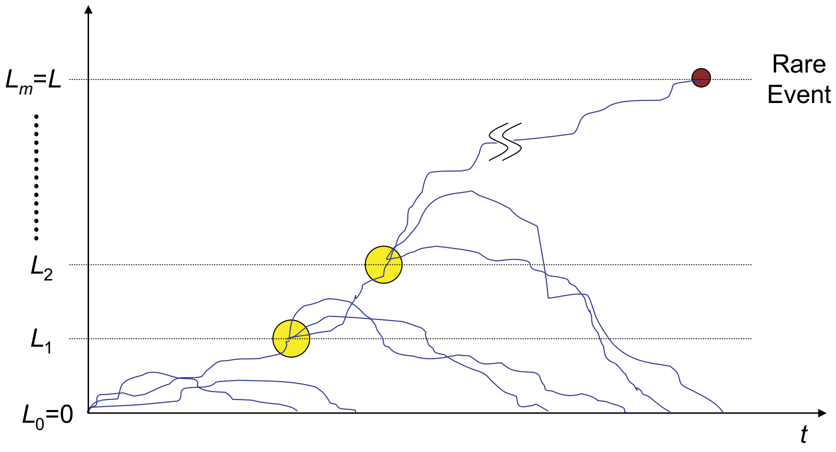


Figure 8.2. Level splitting.

starting point (level 0) after leaving it. Also, let p_i for $i = 1, 2, \ldots, m$, be the conditional probability that the process reaches level i (before returning to level 0) given that the process has reached level $i - 1$. Thus, the rare event probability is

$$\gamma = \prod_{i=1}^{m} p_i.$$

Note that p_i is a conditional probability. The idea of multilevel splitting is to estimate each probability p_i separately, by starting a large number of simulation runs from states at level L_{i-1}.

Let N_i be the number of simulation runs for stage i (from level $i-1$ until either level i or level 0 is reached), and Q_i be the number of runs that reach level i before returning to the initial state (level 0). Then $\hat{p}_i \equiv Q_i/N_i$ is an unbiased estimator of p_i and an unbiased estimator for the rare-event probability γ is

$$\hat{\gamma} = \hat{p}_1 \hat{p}_2 \cdots \hat{p}_m.$$

As N_i increases, $\hat{p}_i$ becomes a better estimation of p_i and so $\hat{\gamma}$ becomes a better estimation of γ. Intuitively, we do not need to *equally* improve the estimation accuracy of $\hat{p}_i$ for different i. Instead, we want to intelligently choose N_i for all i so that the overall simulation efficiency is maximized, i.e., the overall variance is minimized. Specifically, we intend to find the most efficient number of simulation runs for each stage (i.e., N_i, for all i) so that the variance of the rare-event probability estimator $\hat{\gamma}$ is minimized, subject to a computing budget T. The OCBA problem also known as OSTRE (Optimal Splitting Technique for Rare-Event simulation) is

$$\min_{N_1, N_2, \ldots, N_m} \quad \mathrm{Var}(\hat{\gamma}) = \mathrm{Var}(\hat{p}_1 \hat{p}_2 \cdots \hat{p}_m)$$
$$\text{s.t. } c_1 N_1 + c_2 N_2 + \cdots + c_m N_m = T,$$

where c_i is the average one-run simulation cost for stage i. Shortle and Chen [2008] give an asymptotically optimal solution to this computing budget allocation problem as follows.

$$\frac{N_1}{\sqrt{\frac{1-p_1}{c_1 p_1}}} = \frac{N_2}{\sqrt{\frac{1-p_2}{c_2 p_2}}} = \cdots = \frac{N_m}{\sqrt{\frac{1-p_m}{c_m p_m}}}.$$

Further assuming $p_i \ll 1$, a simplified asymptotic solution is

$$N_1 \sqrt{c_1 p_1} = N_2 \sqrt{c_2 p_2} = \cdots = N_m \sqrt{c_m p_m}.$$

Details can be found in Shortle *et al.* [2010].

8.4. Optimal Data Collection Budget Allocation (ODCBA) for Monte Carlo DEA

Data envelopment analysis (DEA) is a mathematical programming approach developed by Charnes *et al.* [1978] to measure efficiency for decision-making units (DMUs) with multiple inputs and multiple outputs. The idea of DEA is to compare different DMUs in terms of how many inputs they have used in achieving the outputs. The most efficient DMUs will be those which use minimum inputs to achieve the maximum outputs. A main advantage of DEA is that it does not require any prior assumptions on the underlying functional relationship between the inputs and outputs [Cooper *et al.* 2006]. A typical DEA model for estimating the efficiency for a DMU (say

DMU j_0) is given as follows.

$$\begin{aligned} \theta(\boldsymbol{X}) = \min \theta & \\ \text{s.t.} \quad & \sum_{j \in J} \lambda_j x_{sj} \le \theta x_{sj_o}, \quad s \in S \\ & \sum_{j \in J} \lambda_j x_{rj} \ge x_{rj_o}, \quad r \in R \\ & \lambda_j \ge 0, \quad j \in J \end{aligned}$$

where S is the set of inputs, R is the set of outputs, J is the set of DMUs, $\boldsymbol{X} = (x_{kj})$, $k \in K$; $j \in J$, consists of all the inputs and outputs for all DMUs, $\theta(\boldsymbol{X})$ is the efficiency score for DMU j_0. Note that the λ_j's are the weights (decision variables) of the inputs/outputs that optimize the efficiency score of DMU j_0. The model above attempts to proportionally contract DMU j_0's inputs as much as possible while not decreasing its current level of outputs. The efficiency score equals one if the DMU is efficient.

One of the requirements for applying DEA is that the values for these input/output variables be known. However these variables are usually unknown in real-life application. Data must be collected from the systems to estimate the distribution of their values. To address this issue, a Monte Carlo sampling method developed by Wong *et al.* [2008] can be applied to estimate the efficiency score. The more data collected, the more accurate these estimations will be.

Given a fixed budget for the data collection, it is important to determine how the data should be collected so that the accuracy of the predicted efficiency score is maximized. The OCBA notion and framework can also be applied to this problem. The objective of this generalized OCBA problem (called ODCBA in this extension) is to minimize the expected mean squared error for the prediction of the efficiency score, subject to a fixed budget for data collection, i.e.,

$$\min_{N_1,\ldots,N_k} E[(\tilde{\theta}(N_1, N_2, \ldots, N_k) - \theta)^2],$$

$$\text{s.t.} \sum_{i=1}^{k} N_i = T.$$

Since there is no closed-form analytical formula for the objective function, a Monte Carlo sampling method is used to estimate the efficiency score. Wong *et al.* [2009] employ a gradient search method to find the optimal solution, i.e., the best data collection scheme. It is shown that the optimal data collection scheme results in a much higher estimation accuracy than the typical data collection scheme which uniformly allocates the data collection budget.

8.5. Other Related Works

There are several other related works along the line of this research. This section gives some examples. By no means, this is an exhaustive list. Instead of finding the design with smallest mean, Trailovic and Pao [2004] develop an OCBA approach for finding a design with minimum variance. They also show a successful application to multi-sensor target tracking problems.

It is well known that positive correlation can help the performance of ranking and selection. Unlike the independence assumption of simulation replications required in this book, Fu *et al.* [2007] extend the OCBA to problems in which the simulation outputs between designs are correlated.

Glynn and Juneja [2004] extend the OCBA to problems in which the simulation output is no longer normally distributed by utilizing large deviation theory. Blanchet *et al.* [2007] further extend the work to heavy-tailed distributions, also utilizing large deviation theory.

Pujowidianto *et al.* [2009] consider constrained optimization problem where the best feasible design is selected from a finite set of alternatives. Both main objective and the constraint measures need to be estimated using simulation. Most other simulation budget allocation approaches either handle only unconstrained problems, or decouple the problem into two phases — first select which designs are feasible and then determine efficient simulation budget allocation to select the best one from the feasible designs. They intend to address the simulation budget allocation problem by considering feasibility and optimality issues at the same time. An asymptotically optimal allocation rule (OCBA-CO) is derived. Numerical experiments indicate that their approach is more efficient than other allocation procedures.

Along the line of efficient simulation allocation for ranking and selection problems, the approach by Chick and Inoue [2001ab] estimates the correct selection probability with Bayesian posterior distributions, and allocates further samples using decision-theory tools to maximize the expected value of information in those samples. Chen and Kelton [2003] made improvement on the traditional indifference-zone selection procedures by incorporating the information of the means for all designs. The procedure by Kim and Nelson [2006] allocates samples in order to provide a guaranteed lower bound for the frequentist probability of correct selection integrated with ideas of early screening. Further, Branke *et al.* [2007] provide a nice overview and extensive comparison for some of the aforementioned selection procedures. They show that OCBA and the VIP approach by Chick and Inoue [2001a] are among the top performers.

To study the impact of sequential simulation allocation, Chen *et al.* [2006] investigate the efficiency gains of using dynamic simulation allocation. Computational results indicate that this sequential version of OCBA, which is based on estimated performance, can easily outperform the optimal static allocation derived using the true sampling distribution. These results imply that the advantage of sequential allocation often outweighs having accurate estimates of the means and variances in determining a good simulation budget allocation. Chick *et al.* [2007] provide a more fundamental discussion about sequential simulation allocation and develop a sequential one-step myopic allocation procedure. Frazier and Powell [2008] consider a Bayesian ranking and selection problem with a correlated multivariate normal prior belief and propose a fully sequential sampling policy called the knowledge-gradient policy.

Appendix A

Fundamentals of Simulation

The purpose of this appendix is to provide some fundamentals related to Monte Carlo simulation and discrete-event simulation. For details, readers can refer to other simulation text books, e.g., Bank *et al.* [2010], or Law and Kelton [2000].

A.1. What is Simulation?

A simulation is an attempt to imitate the operation of a real-life system. It generates an artificial history of the system behavior over time so we can observe, draw inference and answer "what if" questions about the system. Simulation can be used to study the system before it is implemented, and can also be used to understand the current status of the existing system. It can also be used as a tool for predicting the effect of changes in the existing system, and a tool to predict the performance of new systems under a varying set of circumstances.

Simulations can take many forms from spreadsheets to three-dimensional representations of things moving in space. A simulation model can also be continuous or discrete. In the continuous simulation, the state variables changes continuously over time and are usually governed by differential equations. For discrete simulation (also called discrete-event simulation), the state variables only change at a countable number of points in time. These points in time are the times when events occur. A simulation can also be deterministic or stochastic. Stochastic simulation has some uncertain elements in the model, and its behavior cannot be precisely predicted. Whereas in a deterministic model, the output can be precisely predicted if its behavior is understood perfectly.

Simulation is the most appropriate tool to be used when the problem is too complex or difficult to solve using another method. It can be applied to many real world problems, e.g., manufacturing, communication networks, business processes, hospital operations, and container terminal operations. The main advantage of simulation is that it can be used to explore certain behavior of the system without causing disruption in the actual system.

A.2. Steps in Developing A Simulation Model

When we develop a model for a specific system, we need to make sure it captures the various relationships ***existing*** in the current system. Moreover, we need to incorporate some random-number generation procedure so it can generate values for the probabilistic components of the model. A bookkeeping procedure must be developed to keep track of what is happening in the simulation process. Finally as the simulation output is random, certain statistical procedures need to be in place so the accuracy of the output can be guaranteed. Usually the simulation must be conducted for many periods and repeated several times in order to obtain credible results for the decision alternatives or other changes in the system.

In general when we conduct a simulation study, we need the following steps:

1. *Problem identification*

It is always important to identify the right problem before building the model. It should start with understanding the objectives and main purposes of the study. Then based on these objectives, determine what you want to achieve from this simulation study by considering who the stake holder for the study is. This information can help you determine how extensive you want to model the problem, and the scope of the system that you want to model in the simulation.

2. *Model conceptualization*

After the problem is defined, we should determine what components we have to include in the simulation, and also define the interaction between these components. For discrete-event simulation, we need to define the set of events, and how the state changes when an event occurs. This step is also known as model conceptualization. In Section A.2, we will provide more detailed discussion for this topic.

3. *Data collection and input data modeling*

When building the simulation model, we also need to collect data from the system so that they can be used when running the simulation model. For example, when we want to model a fast food restaurant using simulation, we have to collect data on how customers arrive at this restaurant and how fast the customers can be served. Then based on these data, we need to conduct some statistical analysis to choose the distribution that can fit the data the best. This distribution will be used to run the simulation. The statistical analysis is also known as Input Data Modeling. Details will be given in Section A.4.

4. *Model developing*

After defining the conceptual model and conducting input data analysis, we need to translate the model into computer code. In the process of translation, we can either do it using basic computer languages like Basic, FORTRAN, C, C++, or Java, or we can use commercial software such as Arena, Extend, Automod, etc. When coding using a computer language, we have to code the logic of the event scheduler so that the occurrence of the events will be arranged chronologically. Moreover, we also have to code the random number generator so that we can generate data following the certain distributions. Details about the logic behind the event scheduling can be found in Section A.3 while the random number generators can be found in Section A.5. If we are using existing commercial software, we have to code the simulation logic according to the syntax required in the simulation software.

5. Output analysis

For stochastic simulation, it is never enough to only run a single replication. In order to guarantee certain accuracy for the estimate, we need to conduct statistical analysis on the simulation output and then determine how many replications and how long we should run the simulation. Under some cases, we may even need to conduct warm-up analysis so as to make sure there is no bias in our estimate. The details of output analysis are discussed in Section A.6.

6. Validation and verification

We need to verify if the simulation model is accurately coded, and then validate it with the real system to see if it is close to the actual system that we would like to study. We offer further discussions in Section A.7.

The whole simulation studies can be described using Figure A.1. In the following sections, we will give more detailed discussions of the concepts in the simulation model, Input Data Modeling, Output Data Analysis, Random Number Generator, Verification and Validation.

A.3. Concepts in Simulation Model Building

Discrete-event simulation is essentially an interaction of time, events and states. Events are executed chronologically. When an

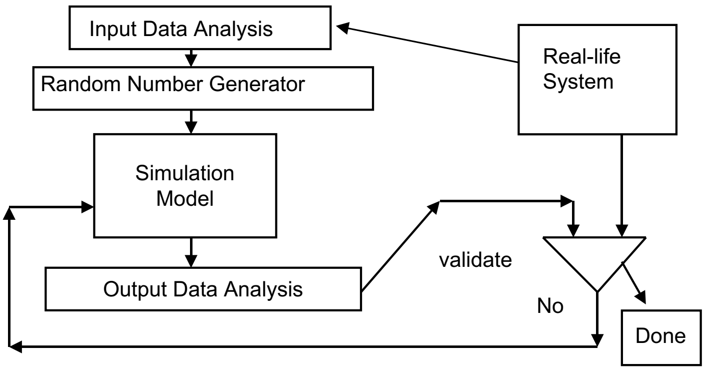


Figure A.1. Steps for simulation study.

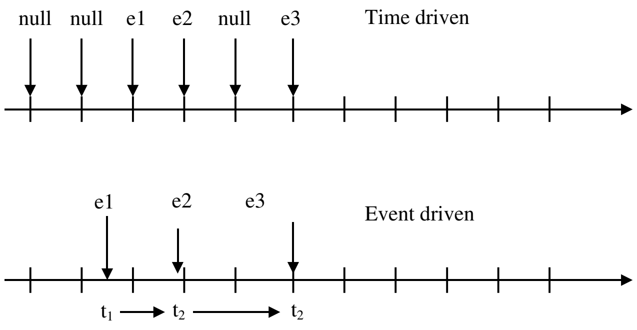


Figure A.2. Events scheduling mechanisms.

event occurs (called triggering event), the states are updated. The triggering event will also trigger the occurrence of some other future events. To ensure the simulation proceeds correctly, we need to have a mechanism which can control the occurrence of the events. There are two mechanisms for advancing simulation time and guaranteeing that all events occur in the correct chronological order. They are time-driven mechanism and event-driven mechanism. The main differences between these two mechanisms are illustrated in Figure A.2. In the time-driven mechanism, the simulation time will move at a constant time step. At every clock tick that the time is advanced, we check if there are any events occurring. It is possible that there are a few events occurring at the same time or there is no event occurring. If there is more than one events occurring, we will execute the events one by one, and the states and the event list will be updated accordingly. If no event takes place, we can treat it as a null event which causes no state change. For the time-driven mechanism, the state transition will be synchronized by the simulation clock.

Event-driven mechanism is a different mechanism. The simulation clock will be advanced to the time when the next event occurs. In this case, only one event is occurring at every time when the simulation clock is advanced.

The event-driven mechanism in general is more efficient than the time-driven mechanism because in the time-driven mechanism, in order to avoid too many events happening at a time click, the time step has to be small, resulting in null events at most of the time clicks. On the other hand, animation will run more smoothly if time proceeds in a constant time-step rather than to the next event directly.

The procedure of the event-driven scheduling mechanism is summarized as follows:

Step 1. Take the first event from the future event list as the triggering event.
Step 2. Advance the simulation time to the time that the event is occurring.
Step 3. Update the state according to the event and the current state.
Step 4. Delete the triggering event and any other infeasible events from the future event list.
Step 5. Add necessary feasible events to the future event list, and sort the events at the future event list according to the event occurring time.
Step 6. Go to Step 1 until the simulation reaches the termination condition.

Example A.1 (G/G/1 simulation). We further explain the event-driven simulation procedure using a G/G/1 simulation example. We first define the events and the states. There are two events in the G/G/1 simulation: one is arrival event, and the other is departure event. The states are LS(t) which is the number of customers at the server (equal to 1 when the server is busy) and LQ(t) which is the number of customers in the queue.

We also need to specify how the events update the state and the event list. When an arrival event occurs, the customer will be sent to the server if the server is idle, else it will join the queue. Moreover, an arrival event will generate another arrival event. Namely, the next customer will be scheduled to arrive after a certain duration of interarrival time. The arrival event will also generate a departure event if the server is idle. Note that if the server is busy, the customer will need to wait in the queue and in this case, we assume the exiting service is not interrupted and so there is no need to change the departure event in the future event list. All the generated events

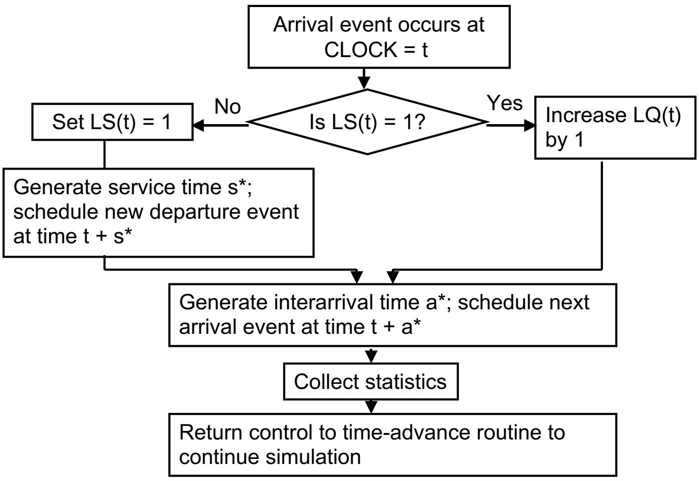

Arrival event occurs at CLOCK = t
Is LS(t) = 1?
No
Yes
Set LS(t) = 1
Increase LQ(t) by 1
Generate service time s*; schedule new departure event at time t + s*
Generate interarrival time a*; schedule next arrival event at time t + a*
Collect statistics
Return control to time-advance routine to continue simulation

Figure A.3. Execution of an arrival event in G/G/1 queue.

will be placed in the future event list. Figure A.3 gives a flow chart showing how an arrival event is executed.

When a departure event occurs, the current service is done and the server will find the next customer in the queue to serve if there are customers in the queue. If no customer is waiting in the queue, the server will become idle. Similarly the departure event will generate another departure event and place it in the future event list if the server is not idle. Figure A.4 gives a flow chart showing how a departure event is executed.

A.4. Input Data Modeling

Input data are needed when running simulations. For example, in the G/G/1 simulation, we need to have the data for the interarrival times and service times. One possible way is to collect data from the actual system and then use them directly in the simulation. However, due to the limited amount of data, this might not be practical. Another way is to generate data based on the information extracted from the collected data. To do this, we fit the collected data using a statistical distribution, and then generate simulation data using this

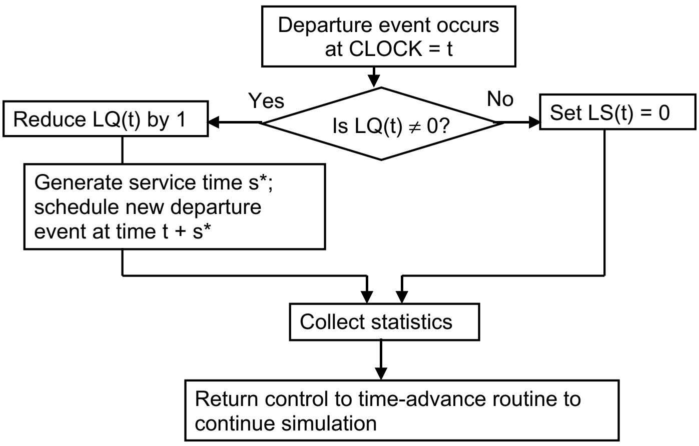

Departure event occurs at CLOCK = t
Is LQ(t) ≠ 0?
Yes
No
Reduce LQ(t) by 1
Set LS(t) = 0
Generate service time s*; schedule new departure event at time t + s*
Collect statistics
Return control to time-advance routine to continue simulation

Figure A.4. Execution of a departure event in G/G/1 queue.

fitted distribution. The process of fitting the data with distributions is known as Input Data Modeling.

There are three steps in doing input data modeling Data Collection, Distribution Fitting, and Goodness of Fit Test.

Data collection

Data collection from the systems is always time-consuming and not trivial. Some important issues to consider when planning the data collection are listed as follows.

- When will be the best time to collect the data?
- What data has to be collected?
- Who should collect the data?
- How to collect the data?

It is advisable to analyze the data while collecting them, as we might need to recollect the data after the analysis. Moreover, be prepared to have surprises from the data, and this can even help you to redefine as well as enhance your model. Draw important insights from the data that you have collected. Sometimes we need to modify the collection plan if it becomes necessary.

Distribution fitting

After the data are collected, we can plot the data using a histogram, From the histogram or the problem nature, we can determine which distribution is the most appropriate. Then we can estimate the parameters for that distribution from the data using some principles such as maximum likelihood estimation or minimum squared errors. Several statistical software packages are available to identify the appropriate distribution as well as its corresponding parameters. Examples include ExpertFit, Arena Input Analyzer, SPSS, SAS, and Eview.

Goodness of fit test

After we have identified the distribution, we still need to check how representative the fitted distribution is. We can do the test using either a heuristic approach or formal statistical tests. For the heuristic approach, we can use a probability plot, and from the plot, we can visually determine if the distribution really fits well with the data. As for statistical tests, we can use either chi-square tests or Kolmogorov–Smirnov tests to check whether the distribution fits the data well. The idea of the Chi-Square Test is to compare the histogram with the probability mass function of the fitted distribution. The Kolmogorov–Smirnov Test focuses on the comparison of the distribution functions between the data and the fitted one.

A.5. Random Number and Variables Generation

When running simulation, we need to generate random numbers that follow some certain distributions for capturing the system's stochastic behaviors, such as service times or customer inter-arrival times. In computer simulation, we use a procedure to generate a sequence of numbers that behave similarly to the random number, and these numbers are called pseudo random numbers. The process that generates these pseudo random numbers is called random number generator.

In order to generate numbers following a certain distribution, we need to first generate a number following a uniform distribution between 0 and 1, and then some random variate generation methods are used to transform the uniform random number into a random number that follows the required distribution. Some details are given in Section A.5.2.

To generate a number following the uniform (0, 1) distribution, we can use the linear congruential method which is presented in Section A.5.1.

A.5.1. *The Linear congruential generators (LCG)*

The linear congruential generator is shown as follows.

$$x_{n+1} = (ax_n + b) \bmod M, \quad \text{and} \tag{A.1}$$

$$u_{n+1} = x_{n+1}/M. \tag{A.2}$$

When a, b and M take appropriate values, u_n will follow a uniform distribution.

Example A.2. Let $a = 2$, $b = 0$, and $M = 16$. Using Equation (A.1) for various x_0, we can get the following sequence of numbers shown in Table A.1. We can see from Table A.1 that no matter what the starting value for x_0 is, the value eventually gets stuck at 0. This implies that the parameters chosen are not good.

Example A.3. Consider a different set of parameter values. Let $a = 11$, $b = 0$ and $M = 16$. Again, we vary the value of the starting point x_0. As shown in Table A.2, we can see that the number stream

Table A.1. Result from a linear congruential generator for Example A.2.

x_0	1	3	5	7	9	11	13	15
x_1	2	6	10	14	18	6	2	14
x_2	4	12	8	12	4	12	4	12
x_3	8	8	0	8	8	8	8	8
x_4	0	0	0	0	0	0	0	0
x_6	0	0	0	0	0	0	0	0

Table A.2. Result from a linear congruential generator for Example A.3.

x_0	1	2	4	5	8	10
x_1	11	6	12	7	8	14
x_2	9	2	4	13	8	10
x_3	3	6	12	15	8	14
x_4	1	2	4	5	8	10
x_6	11	6	12	7	8	14

will repeat itself. Depending on the initial values, some streams have longer periods than the others, where the period is defined as the number of random numbers generated before any number repeats in the sequence.

In general, we need to choose the parameters in such a way that the period is as long as possible because we do not want the numbers to repeat too soon. For the linear congruential generator, the maximum period can be as large as M if we choose the parameters carefully. In addition, the random numbers generated should have the desired properties of uniform distribution. To check whether the generated random number follows a uniform distribution, we can use some statistical tests, for example the chi-square test or Komogorov–Smirnov test. As for testing for independence, we can use the autocorrelation test.

Table A.3 gives four good examples of the parameters suggested in the literature. All of them have $M = 2^{31} - 1 = 2{,}147{,}483{,}647$. The first set has been well used in the C/C++ complier for Unix

Table A.3. Some good choices of the LCG parameters.

a	b	M
1,103,515,245	12,345	$2^{31} - 1$
69,069	97	$2^{31} - 1$
$7^5 = 16{,}807$	0	$2^{31} - 1$
630,360,016	0	$2^{31} - 1$

machines. The third and fourth set have $b = 0$ and so the computational effort of the addition is saved.

A.5.2. *Random variate generation*

There are several methods used for random variate generation with general distributions. We briefly present two most popular methods: Inverse Transform Method and Acceptance Rejection Method.

A.5.2.1. *Inverse transform method*

The idea of the inverse transform method is to generate uniform (0,1) numbers first, and then use the inverse of the distribution function to transform the uniform numbers into the random number which follows the desired distribution. Specifically, let $F(x)$ denote the desired cumulative distribution function and U is uniformly distributed between 0 and 1. Then $X = F^{-1}(U)$ has the distribution function F. The idea is illustrated in Figure A.5. The algorithm is summarized below.

Algorithm (Inverse transform method)

```
Step 1. Generate uniform random numbers $u_1, u_2, \ldots, u_n$;
Step 2. The random numbers that follow the distribution $F(x)$ will
        be $F^{-1}(u_1), F^{-1}(u_2), \ldots, F^{-1}(u_n)$.
```

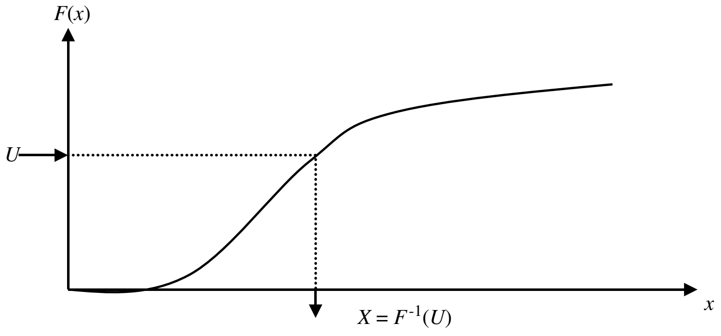


Figure A.5. Inverse transform method in generating random number which follows distribution $F(x)$.

The difficulties of the inverse transform method lie in the derivation for $F^{-1}(u)$. For some distributions, we might not be able to derive the inverse function easily.

Example A.4. For an exponential random variable with rate λ, its cdf is

$$F(u) = 1 - e^{-\lambda u}.$$

Then, its inverse function is

$$F^{-1}(u) = -\ln(1-u)/\lambda.$$

A.5.2.2. *Acceptance rejection method*

The ideas of the acceptance rejection method is to selectively discard samples of the random numbers generated from a distribution (which is usually much easier to generate) so as after the discarding process, the remaining random numbers will follow our intended distribution. We illustrate the idea and present the approach using the following examples.

Example A.5 (Uniform(0.25, 1)). Suppose we want to generate random numbers following the uniform distribution between 0.25 and 1, we can use the following the procedure.

Step 1. Generate a uniform random number u (uniform distribution between 0 and 1),

Step 2. If $u \geq 0.25$, accept $x = u$; else return to Step 1.

Example A.6 (Exponential Thining). We have a stream of random arrivals which follows the Poisson process with rate λ. We can generate a Poisson arrival with rate λ_1 where $\lambda_1 < \lambda$, by randomly discarding p proportion of the arrivals with $p = 1 - \frac{\lambda_1}{\lambda}$.

Example A.7 (Triangular distribution). Consider a random variable X which has the following triangular probability density

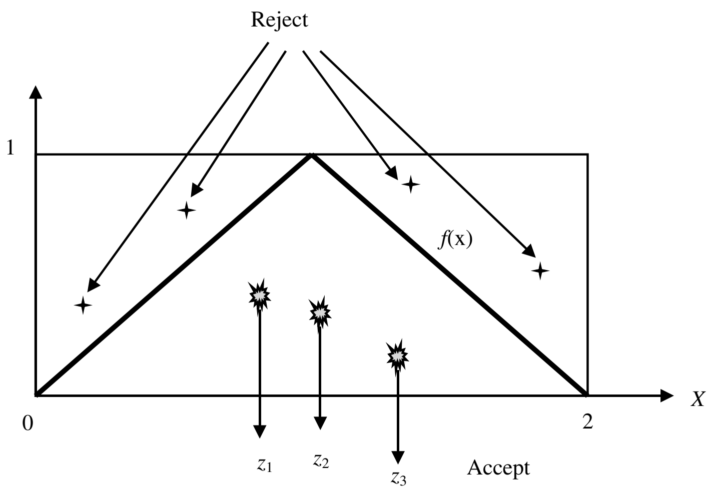


Figure A.6. Acceptance rejection method for a triangular distribution.

function

$$f(x) = \begin{cases} x & 0 \le x \le 1 \\ 2 - x & 1 < x \le 2 \\ 0 & \text{otherwise} \end{cases}$$

We can implement the acceptance rejection method by first generating a uniform number z between 0 and 2, and then another uniform number u from 0 to 1. If the value u is less than or equal to $f(z)$, we will accept the number z, and return $x = z$. Otherwise, we will discard that number and repeat the same procedure until a new number z is accepted. The idea is illustrated as Figure A.6.

The efficiency of this method lies in how many of the numbers that are accepted, i.e., the so-called acceptance rate.

A.6. Output Analysis

Output analysis aims at analyzing the data generated by simulation, and its purpose is to estimate the performance of the system. Since the simulation is stochastic, multiple simulation replications must be performed in order to have a good estimate. The required number of simulation replications depends on the uncertainties of the simulation output. In output analysis, we will look into ways in analyzing the simulation output and then deciding how long we should run for each simulation and how many times we need to replicate the simulation.

Consider the estimation of a performance parameter, $J(\theta)$ of a simulated system θ and we would like to find the point estimate and the interval estimate for $J(\theta)$. There are two different types of output data that we might need to use to estimate $J(\theta)$. One is discrete-time data while the other is continuous-time data. In a queueing system, if the performance we would like to estimate is average waiting time, then the data we use will be the discrete-time data, i.e., the waiting time for each customer. On the other hand, if we want to estimate the average number of customers in the systems, then the data we use will be the continuous-time data, i.e., the number of customers in the system over time.

There are two types of simulation depending on how the simulation is run and how we want it to be terminated. One is terminating simulation and the other is steady-state simulation. For terminating simulation, the condition when the simulation terminates is clearly defined. For example, in the study of customer waiting time at a bank, we know when the simulation stops (i.e., when the bank closes). While for the steady-state simulation, we are interested in the long-run average of the simulation output, or the performance when the system reaches steady state. In theory, the simulation should be run as long as possible. However, in practice, this is not feasible, and so we should determine the conditions under which the simulation can stops, and yet the results obtained are still reasonably well. We will discuss more on this issue in the later part of this section.

Point estimator

The point estimator for $J(\theta)$ based on discrete-time data is the sample mean defined by

$$\bar{J}(\theta) = \frac{1}{n}\sum_{i=1}^{n} L_i(\theta), \tag{A.3}$$

where $\{L_1(\theta), L_2(\theta), \ldots, L_n(\theta)\}$ are the n data that we have collected from the simulation.

The point estimator is unbiased because the expected value of $\bar{J}(\theta)$ is equal to $J(\theta)$. It is always desirable for the estimator to be unbiased.

The point estimator for $J(\theta)$ based on continuous-time data is defined by

$$\bar{J}(\theta) = \frac{1}{T_E}\int_0^{T_E} L(\theta, t)dt, \tag{A.4}$$

where $L(\theta, t)$, $0 \le t \le T_E$, is the output data that we have collected from simulation time 0 to time T_E, where T_E is the time when simulation terminates.

Interval estimator

A point estimator cannot tell users how accurate the estimate is. Hence, an interval estimator is useful in many cases. The interval estimator, also called confidence interval, for $J(\theta)$ is approximately given by

$$\bar{J}(\theta) \pm t_{\alpha/2,d} s_{\bar{J}}(\theta)$$

where $s_{\bar{J}}(\theta)$ is the estimated standard deviation for the point estimator $\bar{J}(\theta)$, and d is the degree of freedom which depends on the amount of data used . The challenge in estimating the confidence interval lies in the estimation of the standard deviation for the point estimator $\bar{J}(\theta)$. This is because to estimate standard deviation, we need to have independent observations. In the following two subsections, we will discuss how to estimate confidence intervals for terminating and steady-state simulations. Moreover, given this estimator, we will discuss how to determine the required number of replications and length of each simulation run.

A.6.1. *Output analysis for terminating simulation*

For terminating simulation, the termination condition of a simulation is well defined. It is determined by the system that we want to study or the objective of the study. Denote N as the number of simulation replications. The confidence interval for $J(\theta)$ is given by

$$\bar{J}(\theta) \pm t_{\alpha/2,N-1} s/\sqrt{N},$$

where $s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(L_i(\theta) - \bar{J}(\theta)\right)^2}$ and $L_i(\theta)$ is the simulation output for the simulation run i. Since the simulation is replicated using independent random number streams, and the independence conditions for $L_i(\theta)$ can be met easily.

To determine how large N should be, we can look at what the desired accuracy level is. If we want a confidence interval whose half width is less than ε, then the number of replications should be

$$N \geq \left(\frac{t_{\alpha/2,N-1} s}{\varepsilon}\right)^2. \tag{A.5}$$

A.6.2. *Output analysis for steady-state simulation*

For steady-state simulation, we want to estimate the steady-state performance. Suppose that the simulation of a single run produces the observations $X_1, X_2, \ldots$. These observations will be correlated. For example the waiting time for a customer will be highly correlated with the waiting time for the customer who comes immediately after this customer. The steady-state performance can be estimated by

$$J(\theta) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} L_i(\theta). \tag{A.6}$$

The estimate will be independent on the initial condition if we have collected infinite amount of observations. However, in practice, due to budget constraints on the computer resources, we will not usually run the simulation very long. Hence the output analysis for steady-state simulation concerns with how long we should run the simulation, and how many replications we should run so that the performance estimate is accurate (or the confidence interval of the estimate is narrow). In general, if we do not want to run the simulation very long, we need to deal with the initial bias. The observations are collected after the system reaches steady state. For example, when we simulate a queueing system, and if we start the simulation with zero customers in the system, the observations that we obtain during the initial stage might be biased (lower than normal as the queue is empty). One approach is to let the simulation run for a while, and only start to collect observations after the system stabilizes. Another alternative is to intelligently set the initial conditions for the simulation so that the simulation can reach the steady state faster. These initial conditions can be estimated by either using the queueing theory or data collected from the real system.

In general, when the steady-state simulation is run, we can divide the simulation into two phases. The first phase is the transient phase (also called warm-up period), and the second phase is the data collection phase. Only the data collected during the data collection phase should be used in estimating the performance.

We can compute the batch means from the data, and it can be used to detect whether the system stabilizes. Batch mean is defined as the average performance within a time interval. For continuous-time data, it equals to the time average of the performance for that interval. For discrete-time data, it equals the average of the observations collected in the interval.

The ensemble average is the average of batch means across different replications. The ensemble average can also be used to detect whether the system stabilizes. However due to the variability of the data, the ensemble average might be too random to observe any trend. Alternatively we can use a moving average or cumulative average to detect if the system stabilizes. After we have determined the time at which the system stabilizes (the length of initial phase), the time the simulation should end is about 10 times longer than the initialization phase.

The formula (A.5) can also be used to determine the number of replications.

A.7. Verification and Validation

After the simulation model is constructed, we have to verify and validate it to ensure it is representative to the actual system. Verification refers to checking the model to see if the conceptual model is accurately coded into a computer program while validation involves with checking the model to see whether it sufficiently represents the actual system so that our objectives of the simulation study can be achieved.

For model verification, we can take the following steps.

1. Build the model in stages. It is always better to start from a simpler model. In every stage of model building, verify if the model is coded accurately in the computer.
2. Ask someone other than the developer to check the computer program.
3. Check the model logic for each action and for each event type.
4. Examine the model output under a variety of settings of the input parameters.
5. Check to see if the input parameters have changed at the end of the simulation.
6. Document the model, and provide precise definition of the variables used.
7. Use animation to check if the simulation behaves similarly to the real system.
8. Use a debugger.
9. Perform stress test, i.e., run the simulation model for a long time and see if the results are reasonable.
10. For certain models, the setting can be changed in a way that close-form solutions for some performance measures exist from queueing theory. For example, there are close-form analytic solution for $M/G/1$ queues or Jackson networks. The theoretical values can be used for verification purpose.

The purpose of validation is to compare the behavior of the simulation model to the real system. However, for validation, we should know that although no model is ever totally representative of the system under study, some models are useful. Hence it is always important to know the objective of the simulation study so that your simulation model is able to help you achieve your objective.

There are two different types of tests we can use for validation.

1. *Subjective test.* It is also known as a qualitative test. We involve people who are knowledgeable on one or more aspects of the system, and let them to check if the model is reasonable.
2. *Objective test.* It is also known as a quantitative test. Statistical tests are used to compare some aspects of the system data set with the model data set. We have to validate the model assumptions, and then compare the model input-output relationships to corresponding input-output relationships in the real system.

Appendix B

Basic Probability and Statistics

This appendix provides some fundamentals of probability and statistics, which will be used in this book. There are many textbooks which give complete coverage of this topic.

B.1. Probability Distribution

There are some distributions which are commonly used in simulation.

Uniform distribution U(a,b)

Suppose the random variable X is uniformly distributed between a and b. We denote it as $X \sim \mathrm{U}(a,b)$. The probability density function (pdf) is

$$f(x) = \begin{cases} \dfrac{1}{b-a}, & \text{if } a \le x \le b \\ 0, & \text{otherwise} \end{cases} .$$

The cumulative distribution function (cdf) is

$$F(x) = \begin{cases} 0, & \text{if } x < a \\ \dfrac{x-a}{b-a}, & \text{if } a \le x \le b \\ 1, & \text{if } x > b \end{cases} .$$

The expected value is

$$E[X] = \frac{a+b}{2},$$

and the variance is

$$Var(X) = \frac{(b-a)^2}{12}.$$

In simulation modeling, if a random quantity is modeled as U(a,b), it implies that this random quantity can be any number between a and b with equal chance but no chance for any number less than a or larger than b.

Triangular distribution

Suppose the random variable X has a triangular distribution between a and b with mode c (the point with highest density). The probability density function (pdf) is

$$f(x) = \begin{cases} \dfrac{2(x-a)}{(b-a)(c-a)}, & \text{if } a \le x \le c \\ \dfrac{2(b-x)}{(b-a)(b-c)}, & \text{if } c \le x \le b \\ 0, & \text{otherwise} \end{cases}.$$

The cumulative distribution function (cdf) is

$$F(x) = \begin{cases} 0, & \text{if } x < a \\ \dfrac{(x-a)^2}{(b-a)(c-a)}, & \text{if } a \le x \le c \\ 1 - \dfrac{(b-x)^2}{(b-a)(b-c)}, & \text{if } c < x \le b \\ 1, & \text{if } b < x \end{cases}.$$

The expected value is

$$E[X] = \frac{a+b+c}{3},$$

and the variance is

$$Var(X) = \frac{a^2+b^2+c^2-ab-ac-bc}{18}.$$

A triangular distribution implies that the modeled random quantity can be any number between a and b but most likely around the mode c. Similar to U(a,b), a triangular distribution has no chance for any number less than a or larger than b.

Exponential distribution

Suppose X is exponentially distributed with mean β, denoted as $X \sim \text{EXP}(\beta)$. The probability density function (pdf) is

$$f(x) = \begin{cases} \dfrac{1}{\beta} e^{-x/\beta}, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}.$$

The cumulative distribution function (cdf) is

$$F(x) = \begin{cases} 1 - e^{-x/\beta}, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}.$$

The expected value is

$$E[X] = \beta,$$

and the variance is

$$Var[X] = \beta^2.$$

Exponential distribution has been well used in stochastic analysis and queueing theory. However, please note that the exponential distribution has the property of "no memory" (or called "memoryless"), i.e.,

$$P\{X > t + t_0 | X > t_0\} = P\{X > t\}.$$

This property is usually true for modeling interarrival times between customers, particularly for uncoordinated arrivals. But the memoryless property could be very wrong in many other cases.

Weibull distribution

The probability density function (pdf) is

$$f(x) = \begin{cases} \dfrac{\alpha}{\beta} \left(\dfrac{x}{\beta}\right)^{\alpha-1} e^{-(x/\beta)^{\alpha}}, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}.$$

The cumulative distribution function (cdf) is

$$F(x) = \begin{cases} 1 - e^{-(x/\beta)^\alpha}, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}.$$

The expected value is

$$E[X] = \beta\,\Gamma(1 + 1/\alpha),$$

and the variance is

$$Var[X] = \beta^2[\Gamma(1 + 2/\alpha) - \Gamma^2(1 + 1/\alpha)],$$

Weibull distribution is suitable for modeling the time to failure of a piece of equipment.

Normal distribution

The probability density function (pdf) is

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}.$$

The expected value is

$$E[X] = \mu,$$

and the variance is

$$Var[X] = \sigma^2.$$

By the principal of the Central Limit Theorem, the normal distribution is suitable to model any performance which is an average in nature.

Poisson distribution

The probability mass function (pmf) is

$$p(x) = \begin{cases} \dfrac{e^{-\lambda}\lambda^x}{x!}, & \text{if } x \in \{0, \ldots\} \\ 0, & \text{otherwise} \end{cases}.$$

The cumulative distribution function (cdf) is

$$F(x) = \begin{cases} 0, & \text{if } x < 0 \\ e^{-\lambda} \displaystyle\sum_{i=0}^{\lfloor x \rfloor} \frac{\lambda^i}{i!}, & \text{if } x \geq 0 \end{cases}.$$

The expected value is

$$E[X] = \lambda,$$

and the variance is

$$Var[X] = \lambda.$$

If the inter-arrival times follow an exponential distribution, the number of customer arrivals in a certain time interval will follow Poisson distribution.

B.2. Some Important Statistical Laws

Law of large number

Let $X_1, X_2, \ldots,$ and X_n be a sequence of independent identically distributed random variables with finite means μ. Their partial sum $S_n = X_1 + X_2 + \cdots + X_n$ satisfies

$$\frac{1}{n} S_n \xrightarrow{D} \mu.$$

Central limit theorem

Let $X_1, X_2, \ldots,$ and X_n be a sequence of independently identically distributed random variables with finite means μ and finite non-zero variance σ^2 and let $S_n = X_1 + X_2 + \cdots + X_n$.

Then

$$\frac{S_n - n\mu}{\sqrt{n\sigma^2}} \xrightarrow{D} N(0,1) \quad \text{as } n \to \infty.$$

B.3. Goodness of Fit Test

Chi-square tests. This is an old but popular test. A chi-square test can be considered as a formal comparison of a histogram with the probability density or mass function of the fitted distribution.

The steps of doing the chi-square test are

Step 1. Draw the frequency histogram of the data.

Step 2. Compute the probability content of each interval, p_i based on the fitted distribution.

Step 3. Compute the expected frequency in each interval $\hat{f}_i = n \times p_i$ where n is the total number of observations.

Step 4. Let f_i be the number of observations in the ith interval. Compute the test statistic:

$$\chi^2 = \sum_i \frac{(\hat{f}_i - f_i)^2}{\hat{f}_i}.$$

Step 5. Determine the critical value at the level of significant α and $\chi^2_{\alpha,d}$ with d degrees of freedom, where $d =$ (number of interval) − (number of estimated parameter) − 1.

Step 6. Conclude that the data does not fit the distribution if the test statistics is greater than the threshold.

Kolmogorov–Smirnov tests. Instead of grouping data in each interval as chi-square tests, Kolmogorov–Smirnov tests intend to compare the empirical distribution function with the fitted distribution ($\hat{F}$). Kolmogorov–Smirnov tests do not require users to group data in any way and so no information is lost. The steps of doing the Kolmogorov–Smirnov test are

Step 1. Rank the data from the smallest to the largest, i.e., $R_{(1)} \leq R_{(2)} \leq \cdots \leq R_{(N)}$.

Step 2. Compute

$$D^+ = \max_{1 \leq i \leq N} \left\{ \frac{i}{N} - \hat{F}(R_{(i)}) \right\},$$

$$D^- = \max_{1 \leq i \leq N} \left\{ \hat{F}(R_{(i)}) - \frac{i-1}{N} \right\}.$$

Step 3. Compute $D = \max(D^+, D^-)$.

Step 4. Determine the critical value, $D_{\alpha,N}$.

Step 5. The data does not fit the distribution if $D > D_{\alpha,N}$.

Appendix C

Some Proofs in Chapter 6

C.1. Proof of Lemma 6.1

We first prove the right-hand side of the inequality, which follows since

$$\psi_i = P\left\{\bigcap_{\substack{j\in\Theta\\ j\neq i}} \big(\tilde{J}_j \not\prec \tilde{J}_i\big)\right\} \le \min_{\substack{j\in\Theta\\ j\neq i}} \big[P\big(\tilde{J}_j \not\prec \tilde{J}_i\big)\big]$$

$$= \min_{\substack{j\in\Theta\\ j\neq i}} \big[1 - P\big(\tilde{J}_j \prec \tilde{J}_i\big)\big] = \min_{\substack{j\in\Theta\\ j\neq i}} \left[1 - \prod_{l=1}^{H} P\big(\tilde{J}_{jl} \le \tilde{J}_{il}\big)\right]. \tag{C.1}$$

To prove the left-hand side of the inequality, we will show that

$$P\left\{\bigcap_{\substack{j\in\Theta\\ j\neq i}} \big(\tilde{J}_j \not\prec \tilde{J}_i\big)\right\} \ge \prod_{\substack{j\in\Theta\\ j\neq i}} P\big(\tilde{J}_j \not\prec \tilde{J}_i\big), \quad i = 1, 2, \ldots, k \tag{C.2}$$

Since then,

$$\psi_i = P\left\{\bigcap_{\substack{j\in\Theta\\ j\neq i}} \big(\tilde{J}_j \not\prec \tilde{J}_i\big)\right\} \ge \prod_{\substack{j\in\Theta\\ j\neq i}} \big[1 - P\big(\tilde{J}_j \prec \tilde{J}_i\big)\big]$$

$$= \prod_{\substack{j\in\Theta\\ j\neq i}} \left[1 - \prod_{l=1}^{H} P\big(\tilde{J}_{jl} \le \tilde{J}_{il}\big)\right], \quad i = 1, 2, \ldots, k.$$

Without loss of generality, we only need to prove the result (C.2) for $i = k$.

For $j = 1, \ldots, k$, let $A_j \equiv (J_{j1}, \ldots, J_{jH})$; and let $A \equiv (A_1, \ldots, A_k)$ and $\tilde{J} \equiv (\tilde{J}_1, \ldots, \tilde{J}_k)$. Define the indicator function

$$I_j(A) \equiv \begin{cases} 0, & \text{if } A_j \prec A_k, \\ 1, & \text{otherwise} \end{cases} \quad \text{for } j = 1, \ldots, k-1. \tag{C.3}$$

Then for $j = 1, \ldots, k-1$, the event

$$\{j \hat{\not\prec} k\} \equiv \{\tilde{J}_j \not\prec \tilde{J}_k\} \quad \text{occurs if and only if } I_j(\tilde{J}) = 1. \tag{C.4}$$

and

$$P\left\{\bigcap_{j=1}^{k-1} (j \not\prec k)\right\} = P\left\{\bigcap_{j=1}^{k-1} (\tilde{J}_j \not\prec \tilde{J}_k)\right\} = E\left[\prod_{j=1}^{k-1} I_j(\tilde{J})\right]. \tag{C.5}$$

We will prove by induction that

$$\begin{aligned} E\left[\prod_{j=1}^{J} I_j(\tilde{J})\right] &\geq \prod_{j=1}^{J} E[I_j(\tilde{J})] \\ &= \prod_{j=1}^{J} P(\tilde{J}_j \not\prec \tilde{J}_k) \quad \text{for } J = 2, \ldots, k-1. \end{aligned} \tag{C.6}$$

Then (C.2) follows immediately for ψ_k. For $J = 2$, we make the following observations:

$$\left.\begin{array}{l} I_1(A) \text{ and } I_2(A) \text{ are both monotone nondecreasing in the single} \\ \text{coordinate (argument) } J_{jl} \text{ for } j = 1, \ldots, k-1 \text{ and } l = 1, \ldots, H; \end{array}\right\} \tag{C.7}$$

and

$$\left.\begin{array}{l} I_1(A) \text{ and } I_2(A) \text{ are both monotone nonincreasing in the single} \\ \text{coordinate (argument) } J_{kl} \text{ for } l = 1, \ldots, H. \end{array}\right\} \tag{C.8}$$

In the terminology of Lehmann [1966], the functions $I_1(A)$ and $I_2(A)$ are concordant in each coordinate J_{jl} because when all the other coordinates are held fixed, both functions $I_1(\cdot)$ and $I_2(\cdot)$ of the single coordinate J_{jl} are monotone in the same direction. Since the components of $\tilde{J} \equiv (\tilde{J}_1, \ldots, \tilde{J}_k)$ are mutually independent random variables, it follows immediately from (C.7), (C.8), and Theorem 1 of Lehmann [1966] that $I_1(\tilde{J})$ and $I_2(\tilde{J})$ are positively quadrant dependent random variables. Then from Lemma 3 of Lehmann [1966], we have

$$E\big[I_1(\tilde{J})\ I_2(\tilde{J})\big] \geq E\big[I_1(\tilde{J})\big]E\big[I_2(\tilde{J})\big]. \tag{C.9}$$

To complete the induction argument, we define

$$I_J^*(A) \equiv \prod_{j=1}^{J} I_j(A) \quad \text{for } J = 2, \ldots, k-1. \tag{C.10}$$

Now suppose by induction that (C.6) holds for some J satisfying $2 \leq J < k-1$. Using the same analysis as in (C.7) and (C.8), we see that

$$\left.\begin{array}{l} I_J^*(A) \text{ and } I_{J+1}^*(A) \text{ are both monotone nondecreasing in the} \\ \text{single coordinate (argument) } J_{jl} \text{ for } j = 1, \ldots, k-1 \\ \text{and } l = 1, \ldots, H \end{array}\right\} \tag{C.11}$$

and

$$\left.\begin{array}{l} I_J^*(A) \text{ and } I_{J+1}^*(A) \text{ are both monotone nonincreasing in the} \\ \text{single coordinate (argument) } J_{kl} \text{ for } l = 1, \ldots, H. \end{array}\right\} \tag{C.12}$$

It follows from Theorem 1 of Lehmann [1966] that $I_J^*(\tilde{J})$ and $I_{J+1}^*(\tilde{J})$ are positively quadrant dependent random variables. (C.13)

From (C.13) and Lemma 3 of Lehmann [1966], we see that

$$\begin{aligned} E\left[\prod_{j=1}^{J+1} I_j(\tilde{J})\right] &= E\big[I_J^*(\tilde{J})I_{J+1}(\tilde{J})\big] \\ &\geq E\big[I_J^*(\tilde{J})\big]E\big[I_{J+1}(\tilde{J})\big] \geq \prod_{j=1}^{J+1} E\big[I_j(\tilde{J})\big] \end{aligned} \tag{C.14}$$

by the induction analysis. This establishes (C.6) and so (C.2) follows immediately. □

C.2. Proof of Lemma 6.2

Since S_p and $\bar{S}_p$ are fixed, we apply the Bonferroni inequality to approximate the two types of errors as follows:

$$\begin{aligned} e_1 &= 1 - P\left\{ \bigcap_{i \in \bar{S}_p} E_i^c \right\} \\ &= 1 - \left(1 - P\left\{ \bigcup_{i \in \bar{S}_p} E_i \right\}\right) \\ &= P\left\{ \bigcup_{i \in \bar{S}_p} E_i \right\} \leq \sum_{i \in \bar{S}_p} P\{E_i\} = \sum_{i \in \bar{S}_p} \psi_i = ae_1, \end{aligned} \tag{C.15}$$

$$\begin{aligned} e_2 &= 1 - P\left\{ \bigcap_{i \in S_p} E_i \right\} \\ &= 1 - \left(1 - P\left\{ \bigcup_{i \in S_p} E_i^c \right\}\right) \\ &= P\left\{ \bigcup_{i \in S_p} E_i^c \right\} \leq \sum_{i \in S_p} P\{E_i^c\} = \sum_{i \in S_p} (1 - \psi_i) = ae_2. \end{aligned} \tag{C.16}$$

□

C.3. Proof of Lemma 6.3

From (C.1) and (C.15), we can obtain

$$\begin{aligned} e_1 \leq ae_1 = \sum_{i \in \bar{S}_p} \psi_i &\leq \sum_{i \in \bar{S}_p} \min_{\substack{j \in \Theta \\ j \neq i}} \left[1 - \prod_{l=1}^{H} P\big(\tilde{J}_{jl} \leq \tilde{J}_{il}\big)\right] \\ &= \sum_{i \in \bar{S}_p} \left[1 - \max_{\substack{j \in \Theta \\ j \neq i}} P\left(\bigcap_{l=1}^{H} \big(\tilde{J}_{jl} \leq \tilde{J}_{il}\big)\right)\right]. \end{aligned} \tag{C.17}$$

By the Bonferroni inequality,

$$P\left(\bigcap_{l=1}^{H}\left(\tilde{J}_{jl} \leq \tilde{J}_{il}\right)\right) \geq \sum_{l=1}^{H} P\big(\tilde{J}_{jl} \leq \tilde{J}_{il}\big) + 1 - H$$
$$\geq H \min_{l \in \{1,\dots,H\}} P\big(\tilde{J}_{jl} \leq \tilde{J}_{il}\big) + 1 - H.$$

After some algebra, we see that an upper bound for Type I error is given by:

$$e_1 \leq ub_1 = H|\bar{S}_p| - H \sum_{i \in \bar{S}_p} \max_{\substack{j \in \Theta \\ j \neq i}} \left(\min_{l \in \{1,\dots,H\}} P\big(\tilde{J}_{jl} \leq \tilde{J}_{il}\big)\right). \quad \text{(C.18)}$$

Similarly, following (C.16), we have

$$e_2 \leq \sum_{i \in S_p} \left(1 - P\Bigg(\bigcap_{\substack{j \in \Theta \\ j \neq i}} (j \,\hat{\nprec}\, i)\Bigg)\right). \quad \text{(C.19)}$$

By the Bonferroni inequality,

$$P\left(\bigcap_{\substack{j \in \Theta \\ j \neq i}} (j \,\hat{\prec}\, i)\right) = 1 - P\left(\bigcup_{\substack{j \in \Theta \\ j \neq i}} (j \hat{\prec} i)\right) \geq 1 - \sum_{\substack{j \in \Theta \\ j \neq i}} P(j \hat{\prec} i)$$
$$\geq 1 - (k-1) \max_{\substack{j \in \Theta \\ j \neq i}} P\left(\bigcap_{l=1}^{H}\left(\tilde{J}_{jl} \leq \tilde{J}_{il}\right)\right).$$

Therefore, after a little algebra, an upper bound for Type II error is

$$e_2 \leq ub_2 = (k-1) \sum_{i \in S_p} \max_{\substack{j \in \Theta \\ j \neq i}} P\left(\bigcap_{l=1}^{H}\left(\tilde{J}_{jl} \leq \tilde{J}_{il}\right)\right)$$
$$\leq (k-1) \sum_{i \in S_p} \max_{\substack{j \in \Theta \\ j \neq i}} \left(\min_{l \in \{1,\dots,H\}} P\big(\tilde{J}_{jl} \leq \tilde{J}_{il}\big)\right). \quad \text{(C.20)}$$

$\square$

C.4. Proof of Lemma 6.5 (Asymptotic Allocation Rules)

C.4.1. *Determination of roles*

To simplify the notation in the proof, we use $k_i = l^i_{j_i}$ instead. From the Larangian function L of the optimization model, we have

$$\begin{aligned}\frac{\partial L}{\partial N_r} &= \frac{\partial APCS_M}{\partial N_r} - \lambda \\ &= H \cdot \sum_{j \notin S_p} \frac{\partial P\big(\tilde{J}_{j_j k_j} \le \tilde{J}_{jk_j}\big)}{\partial N_r} \\ &\quad - (n-1) \cdot \sum_{i \in S_p} \frac{\partial P(\tilde{J}_{j_i k_i} \le \tilde{J}_{ik_i})}{\partial N_r} - \lambda. \qquad \text{(C.21)}\end{aligned}$$

For any design $h \in S_p$,

$$\begin{aligned}\frac{\partial L}{\partial N_h} &= H \cdot \sum_{j \in \Omega_h} \frac{\partial P(\tilde{J}_{hk_j} \le \bar{J}_{jk_j})}{\partial N_h} - (n-1) \cdot \frac{\partial P(\tilde{J}_{j_h k_h} \le \bar{J}_{hk_h})}{\partial N_h} \\ &\quad - (n-1) \cdot \sum_{i \in \Lambda_h} \frac{\partial P(\tilde{J}_{hk_i} \le \tilde{J}_{ik_i})}{\partial N_h} - \lambda, \qquad \text{(C.22)}\end{aligned}$$

where

$$\Omega_h = \{j | j \in \bar{S}_p, j_j = h\}, \qquad \text{(C.23)}$$

$$\Lambda_h = \{i | i \in S_p, j_i = h\}. \qquad \text{(C.24)}$$

By expanding the derivatives, we have

$$\begin{aligned}\frac{\partial L}{\partial N_h} &= -\frac{H}{2\sqrt{2\pi}} \sum_{j \in \Omega_h} \exp\left(-\frac{\delta^2_{jhk_j}}{2\sigma^2_{jhk_j}}\right) \frac{\delta_{jhk_j}\sigma^2_{hk_j}}{\sigma^3_{jhk_j} N_h^2} \\ &\quad + (n-1) \cdot \frac{1}{2\sqrt{2\pi}} \exp\left(-\frac{\delta^2_{hj_hk_h}}{2\sigma^2_{hj_hk_h}}\right) \frac{\delta_{hj_hk_h}\sigma^2_{hk_h}}{\sigma^3_{hj_hk_h} N_h^2} \\ &\quad + (n-1) \cdot \frac{1}{2\sqrt{2\pi}} \sum_{i \in \Lambda_h} \exp\left(-\frac{\delta^2_{ihk_i}}{2\sigma^2_{ihk_i}}\right) \frac{\delta_{ihk_i}\sigma^2_{hk_i}}{\sigma^3_{ihk_i} N_h^2} - \lambda. \end{aligned}$$

(C.25)

Denote

$$C_1 = (n-1) \cdot \frac{1}{2\sqrt{2\pi}} \exp\left(-\frac{\delta^2_{hj_hk_h}}{2\sigma^2_{hj_hk_h}}\right) \frac{\delta_{hj_hk_h}\sigma^2_{hk_h}}{\sigma^3_{hj_hk_h}N_h^2}$$

$$D_1 = (n-1) \cdot \frac{1}{2\sqrt{2\pi}} \sum_{i\in\Lambda_h} \exp\left(-\frac{\delta^2_{ihk_i}}{2\sigma^2_{ihk_i}}\right) \frac{\delta_{ihk_i}\sigma^2_{hk_i}}{\sigma^3_{ihk_i}N_h^2}$$

$$- \frac{H}{2\sqrt{2\pi}} \sum_{j\in\Omega_h} \exp\left(-\frac{\delta^2_{jhk_j}}{2\sigma^2_{jhk_j}}\right) \frac{\delta_{jhk_j}\sigma^2_{hk_j}}{\sigma^3_{jhk_j}N_h^2}$$

Define $\Theta_h = \{i|i \in S, j_i = h\}$, or $\Theta_h = \Lambda_h \cup \Omega_h$. It can be shown that when the conditions

$$\frac{\delta^2_{hj_hk_h}}{\sigma^2_{hj_hk_h}} < \frac{\delta^2_{ihk_i}}{\sigma^2_{ihk_i}}, \quad \forall i \in \Theta_h \tag{C.26}$$

hold, or equivalently when

$$\frac{\delta^2_{hj_hk_h}}{\sigma^2_{hk_h}/\alpha_h + \sigma^2_{j_hk_h}/\alpha_{j_h}} < \min_{i\in\Theta_h} \frac{\delta^2_{ihk_i}}{\sigma^2_{ik_i}/\alpha_i + \sigma^2_{j_hk_i}/\alpha_h}$$

holds, we have

$$\lim_{T\to\infty} \frac{D_1}{C_1} = 0,$$

then (C.25) can be approximated as C_1, and when the condition (C.26) does not hold, we have

$$\lim_{T\to\infty} \frac{C_1}{D_1} = 0$$

then (C.25) can be approximated as D_1.

So we define

$$S_X = \left\{ h\,|h \in S_p,\ \frac{\delta^2_{hj_hk_h}}{\sigma^2_{hk_h}/\alpha_h + \sigma^2_{j_hk_h}/\alpha_{j_h}} < \min_{i\in\Theta_h} \frac{\delta^2_{ihk_i}}{\sigma^2_{ik_i}/\alpha_i + \sigma^2_{hk_i}/\alpha_h} \right\}$$

$$S_Y = S_p \backslash S_X$$

For $h \in S_X$,

$$\frac{\partial L}{\partial N_h} = (n-1) \cdot \frac{1}{2\sqrt{2\pi}} \exp\left(-\frac{\delta^2_{hj_hk_h}}{2\sigma^2_{hj_hk_h}}\right) \frac{\delta_{hj_hk_h}\sigma^2_{hk_h}}{\sigma^3_{hj_hk_h}N_h^2} - \lambda. \tag{C.27}$$

For $h \in S_Y$,

$$\frac{\partial L}{\partial N_h} = (n-1) \cdot \frac{1}{2\sqrt{2\pi}} \sum_{i \in \Lambda_h} \exp\left(-\frac{\delta_{ihk_i}^2}{2\sigma_{ihk_i}^2}\right) \frac{\delta_{ihk_i}\sigma_{hk_i}^2}{\sigma_{ihk_i}^3 N_h^2} - \frac{H}{2\sqrt{2\pi}} \sum_{j \in \Omega_h} \exp\left(-\frac{\delta_{jhk_j}^2}{2\sigma_{jhk_j}^2}\right) \frac{\delta_{jhk_j}\sigma_{hk_j}^2}{\sigma_{jhk_j}^3 N_h^2} - \lambda. \quad \text{(C.28)}$$

Similarly, for any design $r \in \bar{S}_p$, we define

$$S_U = \left\{ r \,\middle|\, r \in \bar{S}_p,\ \frac{\delta_{rj_rk_r}^2}{\sigma_{rk_r}^2/\alpha_r + \sigma_{j_rk_r}^2/\alpha_{j_r}} < \min_{i \in \Theta_r} \frac{\delta_{irk_i}^2}{\sigma_{ik_i}^2/\alpha_i + \sigma_{rk_i}^2/\alpha_r} \right\}, \quad \text{(C.29)}$$

$$S_V = \bar{S}_p \backslash S_U \quad \text{(C.30)}$$

and we have the following approximations:

For $r \in S_U$,

$$\frac{\partial L}{\partial N_r} = -\frac{H}{2\sqrt{2\pi}} \exp\left(-\frac{\delta_{rj_rk_r}^2}{2\sigma_{rj_rk_r}^2}\right) \frac{\delta_{rj_rk_r}\sigma_{rk_r}^2}{\sigma_{rj_rk_r}^3 N_r^2} - \lambda. \quad \text{(C.31)}$$

For $r \in S_V$,

$$\frac{\partial L}{\partial N_r} = (n-1) \cdot \frac{1}{2\sqrt{2\pi}} \sum_{i \in \Lambda_r} \exp\left(-\frac{\delta_{irk_i}^2}{2\sigma_{irk_i}^2}\right) \frac{\delta_{irk_i}\sigma_{rk_i}^2}{\sigma_{irk_i}^3 N_r^2} - \frac{H}{2\sqrt{2\pi}} \sum_{j \in \Omega_r} \exp\left(-\frac{\delta_{jrk_j}^2}{2\sigma_{jrk_j}^2}\right) \frac{\delta_{jrk_j}\sigma_{rk_j}^2}{\sigma_{jrk_j}^3 N_r^2} - \lambda. \quad \text{(C.32)}$$

It is observed that determination of subsets S_X and S_U shares the same condition and the two subsets can be merged, which is the same case for S_Y and S_V. Thus we redefine

$$S_A = \left\{ h \,\middle|\, h \in S,\ \frac{\delta_{hj_hk_h}^2}{\sigma_{hk_h}^2/\alpha_h + \sigma_{j_hk_h}^2/\alpha_{j_h}} < \min_{i \in \Theta_h} \frac{\delta_{ihk_i}^2}{\sigma_{ik_i}^2/\alpha_i + \sigma_{j_hk_i}^2/\alpha_h} \right\}, \quad \text{(C.33)}$$

$$S_B = S \backslash S_A, \quad \text{(C.34)}$$

where

$$\Theta_h = \{i | i \in S, j_i = h\}.$$

C.4.2. *Allocation rules*

C.4.2.1. $h \in S_A$, $o \in S_A$

There are 4 possible cases for h and o, which are

1. $h \in S_X$, $o \in S_X$
2. $h \in S_X$, $o \in S_U$
3. $h \in S_U$, $o \in S_U$
4. $h \in S_U$, $o \in S_X$

For case 1, we have

$$\frac{\partial L}{\partial N_h} = (n-1) \cdot \frac{1}{2\sqrt{2\pi}} \exp\left(-\frac{\delta^2_{hj_hk_h}}{2\sigma^2_{hj_hk_h}}\right) \frac{\delta_{hj_hk_h}\sigma^2_{hk_h}}{\sigma^3_{hj_hk_h}N_h^2} - \lambda$$

$$\frac{\partial L}{\partial N_o} = (n-1) \cdot \frac{1}{2\sqrt{2\pi}} \exp\left(-\frac{\delta^2_{oj_ok_o}}{2\sigma^2_{oj_ok_o}}\right) \frac{\delta_{oj_ok_o}\sigma^2_{ok_o}}{\sigma^3_{oj_ok_o}N_o^2} - \lambda$$

Since

$$\frac{\partial L}{\partial N_h} = \frac{\partial L}{\partial N_o} = 0$$

then

$$\exp\left(-\frac{\delta^2_{hj_hk_h}}{2\sigma^2_{hj_hk_h}}\right) \frac{\delta_{hj_hk_h}\sigma^2_{hk_h}}{\sigma^3_{hj_hk_h}N_h^2} = \exp\left(-\frac{\delta^2_{oj_ok_o}}{2\sigma^2_{oj_ok_o}}\right) \frac{\delta_{oj_ok_o}\sigma^2_{ok_o}}{\sigma^3_{oj_ok_o}N_o^2}$$

or

$$\exp\left(\frac{\delta^2_{oj_ok_o}}{2\sigma^2_{oj_ok_o}} - \frac{\delta^2_{hj_hk_h}}{2\sigma^2_{hj_hk_h}}\right) = \frac{\alpha_h^2\sigma^3_{hj_hk_h}}{\alpha_o^2\sigma^3_{oj_ok_o}} \frac{\delta_{oj_ok_o}\sigma^2_{ok_o}}{\delta_{hj_hk_h}\sigma^2_{hk_h}}, \qquad \text{(C.35)}$$

where substitution $N_i = \alpha_i T$ is taken if necessary.

Since $\sigma^2_{ijk} = \frac{\sigma^2_{ik}}{N_i} + \frac{\sigma^2_{jk}}{N_j} = \frac{\sigma^2_{ik}}{\alpha_i T} + \frac{\sigma^2_{jk}}{\alpha_j T}$, and denote

$$\rho_i = \frac{N_{j_i}}{N_i} = \frac{\alpha_{j_i}}{\alpha_i}, \qquad \text{(C.36)}$$

then

$$\sigma^2_{ij_ik_i} = \frac{\sigma^2_{ik_i}}{\alpha_i T} + \frac{\sigma^2_{j_ik_i}}{\alpha_{j_i} T}. \qquad \text{(C.37)}$$

Take natural log on both sides of (C.35)

$$\frac{\delta^2_{oj_ok_o}\alpha_o}{\sigma^2_{ok_o}+\sigma^2_{j_ok_o}/\rho_o}-\frac{\delta^2_{hj_hk_h}\alpha_h}{\sigma^2_{hk_h}+\sigma^2_{j_hk_h}/\rho_h}$$
$$=\frac{2}{T}\left(\ln\frac{\alpha_h^{\frac{1}{2}}\left(\sigma^2_{hk_h}+\sigma^2_{j_hk_h}/\rho_h\right)^{\frac{3}{2}}\delta_{oj_ok_o}\sigma^2_{ok_o}}{\alpha_o^{\frac{1}{2}}\left(\sigma^2_{ok_o}+\sigma^2_{j_ok_o}/\rho_o\right)^{\frac{3}{2}}\delta_{hj_hk_h}\sigma^2_{hk_h}}\right)\quad\text{(C.38)}$$

when $T\to\infty$, the limit of right hand side of (C.38) is 0. Thus

$$\frac{\delta^2_{oj_ok_o}\alpha_o}{\sigma^2_{ok_o}+\sigma^2_{j_ok_o}/\rho_o}=\frac{\delta^2_{hj_hk_h}\alpha_h}{\sigma^2_{hk_h}+\sigma^2_{j_hk_h}/\rho_h}$$

which implies

$$\frac{\alpha_h}{\alpha_o}=\frac{\left(\sigma^2_{hk_h}+\sigma^2_{j_hk_h}/\rho_h\right)/\delta^2_{hj_hk_h}}{\left(\sigma^2_{ok_o}+\sigma^2_{j_ok_o}/\rho_o\right)/\delta^2_{oj_ok_o}}.\quad\text{(C.39)}$$

For case 2, 3 and 4, following similar derivations, it can be shown that the above rule also holds.

C.4.2.2. $d\in S_B$

There are 2 cases for d, which are

1. $d\in S_Y$
2. $d\in S_V$

For case 1, from (C.8), we have

$$\frac{\partial L}{\partial N_d}=(n-1)\cdot\frac{1}{2\sqrt{2\pi}}\sum_{i\in\Lambda_d}\exp\left(-\frac{\delta^2_{idk_i}}{2\sigma^2_{idk_i}}\right)\frac{\delta_{idk_i}\sigma^2_{dk_i}}{\sigma^3_{idk_i}N_d^2}$$
$$-\frac{H}{2\sqrt{2\pi}}\sum_{j\in\Omega_d}\exp\left(-\frac{\delta^2_{jdk_j}}{2\sigma^2_{jdk_j}}\right)\frac{\delta_{jdk_j}\sigma^2_{dk_j}}{\sigma^3_{jdk_j}N_d^2}-\lambda$$
$$=(n-1)\cdot\frac{1}{2\sqrt{2\pi}}\sum_{h\in\Lambda_d^*}\exp\left(-\frac{\delta^2_{hdk_h}}{2\sigma^2_{hdk_h}}\right)\frac{\delta_{hdk_h}\sigma^2_{dk_h}}{\sigma^3_{hdk_h}N_d^2}$$
$$-\frac{H}{2\sqrt{2\pi}}\sum_{r\in\Omega_d^*}\exp\left(-\frac{\delta^2_{rdk_r}}{2\sigma^2_{rdk_r}}\right)\frac{\delta_{rdk_r}\sigma^2_{dk_r}}{\sigma^3_{rdk_r}N_d^2}-\lambda$$

where

$$\Lambda_d^* = \{h | h \in S_A, j_h = d\}, \tag{C.40}$$

$$\Omega_d^* = \{r | r \in S_U, j_r = d\}. \tag{C.41}$$

Since

$$\frac{\partial L}{\partial N_h} = (n-1) \cdot \frac{1}{2\sqrt{2\pi}} \exp\left(-\frac{\delta_{hdk_h}^2}{2\sigma_{hdk_h}^2}\right) \frac{\delta_{hdk_h}\sigma_{hk_h}^2}{\sigma_{hdk_h}^3 N_h^2} - \lambda$$

$$\frac{\partial L}{\partial N_r} = -\frac{H}{2\sqrt{2\pi}} \exp\left(-\frac{\delta_{rdk_r}^2}{2\sigma_{rdk_r}^2}\right) \frac{\delta_{rdk_r}\sigma_{rk_r}^2}{\sigma_{rdk_r}^3 N_r^2} - \lambda$$

and

$$\frac{\partial L}{\partial N_d} = \frac{\partial L}{\partial N_h} = \frac{\partial L}{\partial N_r} = 0,$$

then by substitution we can get

$$N_d^2 = \sum_{h \in \Lambda_d^*} \frac{\sigma_{dk_h}^2}{\sigma_{hk_h}^2} N_h^2 + \sum_{r \in \Omega_d^*} \frac{\sigma_{dk_r}^2}{\sigma_{rk_r}^3} N_r^2$$

substituting $N_i = \alpha_i T$, we get

$$\alpha_d^2 = \sum_{h \in \Theta_d^*} \frac{\sigma_{dk_h}^2}{\sigma_{hk_h}^2} \alpha_h^2$$

where

$$\Theta_d^* = \{h | h \in S_A, j_h = d\}. \tag{C.42}$$

For case 2, following a similar derivation, it can be proven that the allocation rule above still holds.

Appendix D

Some OCBA Source Codes

There are several source codes for the OCBA algorithms presented in this book, which can be downloaded from the OCBA web sites:

http://volgenau.gmu.edu/~cchen9/ocba.html
or
http://en.wikipedia.org/wiki/Optimal_Computing_Budget_Allocation

This appendix only gives the computer code for the OCBA method presented in Chapter 3 and 4. This OCBA algorithm is to find the single best design using a minimum simulation budget. The code runs on C or C++ platform.

This code is a self-contained program which contains the "ocba" subroutine and a main program. The main program calls the OCBA subroutine and intends to allocate a computing budget of 100 additional simulation runs. If it runs successfully, it should output the following:

```
The additional computation budget assigned to Design 0 is 48.
The additional computation budget assigned to Design 1 is 38.
The additional computation budget assigned to Design 2 is 11.
The additional computation budget assigned to Design 3 is 2.
The additional computation budget assigned to Design 4 is 1.
```

C/C++ Code for Optimal Computing Budget Allocation (OCBA)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* please put the following definitions at the head of your
     program */
/*Define parameters for simulation*/
/*the number of designs. It should be larger than 2.*/
#define ND 5

/*prototypes of functions and subroutines included in this
file*/
void ocba();
int best();
int second_best();

void main()
    {
    int i;
    int ADD_BUDGET=100;      /*the additional simulation
budget. It should be positive integer.*/
    float s_mean[ND]={1.2, 2.1, 3.4, 4.87, 6.05};
    float s_var[ND]= {3.3, 2.0, 4.5, 5.3, 6.9};
    int n[ND]={12,6,5,5,4};
    int an[ND];

    ocba(s_mean, s_var, ND, n, ADD_BUDGET, an, 1);

    for(i=0;i<ND;i++)
        {
        printf("The additional computation budget assigned to
Design %d is %ld.\n", i, an[i]);
        }
    }


void ocba(s_mean,s_var,nd,n,add_budget,an,type)
float *s_mean,*s_var;
```

```
int nd,*n,add_budget,*an,type;
/*
This subroutine determines how many additional runs each
design will should have for next iteration of simulation.
   s_mean[i]: sample mean of design i, i=0,1,..,ND-1
   s_var[i]: sample variance of design i, i=0,1,..,ND-1
   nd: the number of designs
   n[i]: number of simulation replications of design i,
         i=0,1,..,ND-1
   add_budget: the additional simulation budget
   an[i]: additional number of simulation replications assigned
          to design i, i=0,1,..,ND-1
   type: type of optimazition problem. type=1, MIN problem;
type=2, MAX problem
*/
    {
    int i,j;
    int b, s;
    int t_budget,t1_budget;
        int morerun[ND],more_alloc; /* 1:Yes; 0:No */
    float t_s_mean[ND];
    float ratio[ND]; /* Ni/Ns */
    float ratio_s,temp;

    if(nd>ND) printf("\n !!!!! please stop the program and
increase ND");

    if (type == 1) /*MIN problem*/
      {
      for(i=0; i<nd; i++) t_s_mean[i] = s_mean[i];
      }
    else  /*MAX problem*/
      {
      for(i=0; i<nd; i++) t_s_mean[i] = (-1)*s_mean[i];
      }

    t_budget=add_budget;
    for(i=0; i<nd; i++) t_budget+=n[i];
    b=best(t_s_mean, nd);
    s=second_best(t_s_mean, nd, b);
      ratio[s]=1.0;
    for(i=0;i<nd;i++)
```

```
       if(i!=s && i!=b)
         {
             temp=(t_s_mean[b]-t_s_mean[s])/(t_s_mean[b]-
       t_s_mean[i]);
             ratio[i]=temp*temp*s_var[i]/s_var[s];
             }    /* calculate ratio of Ni/Ns*/
         temp=0;
         for(i=0;i<nd;i++) if(i!=b) temp+=(ratio[i]*ratio[i]/
       s_var[i]);
         ratio[b]=sqrt(s_var[b]*temp); /* calculate Nb */
      for(i=0;i<nd;i++) morerun[i]=1;
      t1_budget=t_budget;
      do{
        more_alloc=0;
        ratio_s=0.0;
        for(i=0;i<nd;i++) if(morerun[i]) ratio_s+=ratio[i];
        for(i=0;i<nd;i++) if(morerun[i])
          {
          an[i]=(int)(t1_budget/ratio_s*ratio[i]);
      /* disable those design which have been run too much */
          if(an[i]<n[i])
            {
            an[i]=n[i];
            morerun[i]=0;
            more_alloc=1;
            }
          }
        if(more_alloc)
          {
          t1_budget=t_budget;
          for(i=0;i<nd;i++) if(!morerun[i]) t1_budget-=an[i];
          }
        } while(more_alloc); /* end of WHILE */
                            /* calculate the difference */
        t1_budget=an[0];
        for(i=1;i<nd;i++) t1_budget+=an[i];
        an[b]+=(t_budget-t1_budget);     /* give the difference
      to design b */
        for(i=0;i<nd;i++) an[i]-=n[i];
}
```

```
int best(t_s_mean,nd)
float *t_s_mean;
int nd;
/*This function determines the best design based on current
simulation results */
/* t_s_mean[i]: temporary array for sample mean of design i,
i=0,1,..,ND-1
   nd: the number of designs */
    {
    int i, min_index;
    min_index=0;
    for(i=0;i<nd;i++)
        {
        if(t_s_mean[i]<t_s_mean[min_index])
            {
            min_index=i;
            }
        }
    return min_index;
    }


int second_best(t_s_mean,nd,b)
float *t_s_mean;
int nd;
int b;
/*This function determines the second best design based on
current simulation results*/
/* t_s_mean[i]: temporary array for sample mean of design i,
i=0,1,..,ND-1
   nd: the number of designs.
   b: current best design determined by function
best() */
    {

    int i, second_index;
    if(b==0) second_index=1;
    else second_index=0;
    for(i=0;i<nd;i++)
```

```
        {
        if(t_s_mean[i]<t_s_mean[second_index] && i!=b)
            {
            second_index=i;
            }
        }
    return second_index;
    }
```

References

Banks, J., Carson II, J. S., Nelson, B. L. and Nicol, D. M. [2004] *Discrete-Event System Simulation*, 4th edn. (Prentice Hall).

Banks, J., Carson II, J. S., Nelson, B. L. and Nicol, D. M. [2010] *Discrete Event-System Simulation*, 5th edn. (Prentice Hall).

Barton, R. R. and Meckesheimer, M. [2006] "Metamodel-based simulation optimization," in *Handbooks in Operations Research and Management Science: Simulation*, eds. Henderson, S. G. and Nelson, B. L. (Elsevier) Chapter 18, pp. 535–574.

Bechhofer, R. E., Santner, T. J. and Goldsman, D. M. [1995] *Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons* (John Wiley & Sons).

Berger, P. D. and Maurer, R. E. [2002] *Experimental Design with Applications in Management, Engineering, and the Sciences* (Duxbury).

Bernardo, J. M. and Smith, A. F. M. [1994] *Bayesian Theory* (Wiley, New York).

Blanchet, J., Liu, J. and Zwart, B. [2007] "Large deviations perspective on ordinal optimization of heavy-tailed systems," *Proc. 2007 Winter Simulation Conf.*, pp. 489–494.

Branke, J., Chick, S. E. and Schmidt, C. [2007] "Selecting a selection procedure," *Manag. Sci.* **53**, 1916–1932.

Brantley, M. W., Lee, L. H., Chen, C. H. and Chen, A. [2008] "Optimal sampling in design of experiment for simulation-based stochastic optimization," *Proc. 2008 IEEE Conf. on Automation Science and Engineering*, Washington, DC, pp. 388–393.

Bratley, P., Fox, B. L. and Schrage, L. E. [1987] *A Guide to Simulation*, 2nd ed. (Springer-Verlag).

Butler, J., Morrice, D. J. and Mullarkey, P. W. [2001] "A multiple attribute utility theory approach to ranking and selection," *Manag. Sci.* **47**(6), 800–816.

Chen, C. H. [1995] "An effective approach to smartly allocate computing budget for discrete event simulation," *Proc. of the 34th IEEE Conf. on Decision and Control*, pp. 2598–2605.

Chen, C. H. [1996] "A lower bound for the correct subset-selection probability and its application to discrete event system simulations," *IEEE Trans. Automat. Contr.* **41**, 1227–1231.

Chen, H. C., Chen, C. H., Dai, L. and Yücesan, E. [1997] "New development of optimal computing budget allocation for discrete event simulation," *Proc. 1997 Winter Simulation Conf.*, pp. 334–341.

Chen, C. H., Chen, H. C., Yücesan, E. and Dai, L. [1998] "Computing budget allocation for simulation experiments with different system structures," *Proc. 1998 Winter Simulation Conf.*, pp. 735–741.

Chen, C. H., Fu, M. and Shi, L. [2008] "Simulation and Optimization," in *Tutorials in Operations Research* (Informs, Hanover, MD), pp. 247–260.

Chen, C. H. and He, D. [2005] "Intelligent simulation for alternatives comparison and application to air traffic management," *J. Syst. Sci. Syst. Eng.* **14**(1), 37–51.

Chen, C. H., He, D. and Fu, M. [2006] "Efficient dynamic simulation allocation in ordinal optimization," *IEEE Trans. Automat. Contr.* **51**(12), 2005–2009.

Chen, C. H., He, D., Fu, M. and Lee, L. H. [2008] "Efficient simulation budget allocation for selecting an optimal subset," *Info. J. Comput.* **20**(4), 579–595.

Chen, C. H., Kumar, V. and Luo, Y. C. [1999] "Motion planning of walking robots in environments with uncertainty," *J. Robot. Syst.* **16**(10), 527–545.

Chen, C. H., Lin, J., Yücesan, E. and Chick, S. E. [2000] "Simulation budget allocation for further enhancing the efficiency of ordinal optimization," *J. Discrete Event Dynam. Syst. Theor. Appl.* **10**, 251–270.

Chen, C. H., Wu, S. D. and Dai, L. [1999] "Ordinal comparison of heuristic algorithms using stochastic optimization," *IEEE Trans. Robot. Automat.* **15**(1), 44–56.

Chen, C. H. and Yücesan, E. [2005] "An alternative simulation budget allocation scheme for efficient simulation," *Int. J. Simulat. Process Model.* **1**(1/2), 49–57.

Chen, C. H., Yücesan, E., Dai, L. and Chen, H. C. [2010] "Efficient computation of optimal budget allocation for discrete event simulation experiment," *IIE Trans.* **42**(1), 60–70.

Chen, E. J. and Kelton, W. D. [2003] "Inferences from indifference-zone selection procedures," *Proc. 2003 Winter Simulation Conf.*, pp. 456–464.

Cheng, R. C. H. [1982] "Antithetic variate methods for simulation of processes with peaks and troughs," *Eur. J. Oper. Res.* **33**, 229–237.

Cheng, R. C. H. [1984] "The use of antithetic variates in computer simulations," *J. Oper. Res. Soc.* **15**, 227–236.

Chew, E. P., Lee, L. H., Teng, S. Y. and Koh, C. H. [2009] "Differentiated service inventory optimization using nested partitions and MOCBA," *Comput. Oper. Res.* **36**(5), 1703–1710.

Chick, S. E. [1997] "Selecting the best system: A decision-theoretic approach," *Proc. 1997 Winter Simulation Conf.*, pp. 326–333.

Chick, S. E. [2003] "Expected opportunity cost guarantees and indifference-zone selection procedures," *Proc. 2003 Winter Simulation Conf.*, pp. 465–473.

Chick, S. E., Branke, J. and Schmidt, C. [2007] "New greedy myopic and existing asymptotic sequential selection procedures: Preliminary empirical results," *Proc. 2007 Winter Simulation Conf.*, pp. 289–296.

Chick, S. and Inoue, K. [2001a] "New two-stage and sequential procedures for selecting the best simulated system," *Oper. Res.* **49**, 1609–1624.

Chick, S. and Inoue, K. [2001b] "New procedures to select the best simulated system using common random numbers," *Manag. Sci.* **47**, 1133–1149.

Chick, S. E. and Gans, N. [2006] "Simulation selection problems: Overview of an economic analysis," *Proc. 2006 Winter Simulation Conf.*, pp. 279–286.

Dai, L. and Chen, C. H. [1997] "Rate of convergence for ordinal comparison of dependent simulations in discrete event dynamic systems," *J. Optim. Theor. Appl.* **94**(1), 29–54.

Dai, L., Chen, C. H. and Birge, J. R. [2000] "Large convergence properties of two-stage stochastic programming," *J. Optim. Theor. Appl.* **106**(3), 489–510.

DeGroot, M. H. [1970] *Optimal Statistical Decisions* (McGraw-Hill, New York).

Dudewicz, E. J. and Dalal, S. R. [1975] "Allocation of observations in ranking and selection with unequal variances," *Sankhya* **B37**, 28–78.

Frazier, P. and Powell, W. B. [2008] "The knowledge-gradient stopping rule for ranking and selection," *Proc. 2008 Winter Simulation Conf.*, pp. 305–312.

Fu, M. [2002] "Optimization for simulation: Theory vs. practice," *Info. J. Comput.* **14**(3), 192–215.

Fu, M. C. [2006] "Gradient estimation," in *Handbooks in Operations Research and Management Science: Simulation*, eds. Henderson, S. G. and Nelson, B. L. (Elsevier), Chapter 19, pp. 575–616.

Fu, M. C. [2007] "Are we there yet? the marriage between simulation & optimization," *OR/MS Today*, pp. 16–17.

Fu, M. C., Hu, J. Q., Chen, C. H. and Xiong, X. [2007] "Simulation allocation for determining the best design in the presence of correlated sampling," *Info. J. Comput.* **19**(1), 101–111.

Fu, M. C., Hu, J. and Marcus, S. I. [2006] "Model-based randomized methods for global optimization," *Proc. 17th Int. Symp. on Mathematical Theory of Networks and Systems*, 355–363.

Fu, M, Chen, C. H. and Shi, L. [2008] "Some topics for simulation optimization," *Proc. 2008 Winter Simulation Conf.*, Miami, FL, pp. 27–38.

Fu, M. C. and Hu, J. Q. [1997] *Conditional Monte Carlo: Gradient Estimation and Optimization Applications* (Kluwer Academic Publishers).

Fonseca, C. M. and Fleming, P. J. [1995] "An overview of evolutionary algorithms in multiobjective optimization," *Evol. Comput.* **3**(1), 1–16.

Garvels, M. J. J., Kroese, D. P. and Van Ommeren, J.-K. C. W. [2002] "On the importance function in splitting simulation," *Eur. Trans. Telecommunications* **13**(4), 363–371.

Glasserman, P. [1991] *Gradient Estimation via Perturbation Analysis* (Kluwer Academic, Boston, Massachusetts).

Glasserman, P., Heidelberger, P., Shahabuddin, P. and Zajic, T. [1999] "Multilevel splitting for estimating rare event probabilities," *Oper. Res.* **47**(4), 585–600.

Glynn, P. W. [1994] "Efficiency improvement technique," *Annals of Operations Research.*

Glynn, P. and Juneja, S. [2004] "A large deviations perspective on ordinal optimization," *Proc. 2004 Winter Simulation Conf.*, pp. 577–585.

Goldberg, D. E. [1989] *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley).

Goldsman, D. and Nelson, B. L. [1998] "Statistical screening, selection, and multiple comparisons in computer simulation," *Proc. Winter Simulation Conf.*, pp. 159–166.

Gupta, S. S. [1965] "On some multiple decision (selection and ranking) rules," *Technometrics* **7**, 225–245.

Hanne, T. and Nickel, S. [2005] "A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects," *Eur. J. Oper. Res.* **167**, 663–678.

He, D., Chick, S. E. and Chen, C. H. [2007] "The opportunity cost and OCBA selection procedures in ordinal optimization," *IEEE Trans. on Systems, Man, and Cybernetics — Part C* **37**(5), 951–961.

He, D., Lee, L. H., Chen, C. H., Fu, M. and Wasserkrug, S. [2010] "Simulation optimization using the cross-entropy method with optimal computing budget allocation," *ACM Trans. on Modeling and Computer Simulation* **20**(1), 1–22.

Heidelberger, P. [1993] "Fast simulation of rare events in queueing and reliability models," in *Performance Evaluation of Computer and Communication Systems*, eds. Donatiello, L. and Nelson, R. (Springer Verlag), pp. 165–202.

Ho, Y. C. and Cao, X. R. [1991] *Perturbation Analysis and Discrete Event Dynamic Systems* (Kluwer Academic).

Ho, Y. C., Cassandras, C. G., Chen, C. H. and Dai, L. [2000] "Ordinal optimization and simulation," *J. Oper. Res. Soc.* **51**(4), 490–500.

Ho, Y. C., Sreenivas, R. S. and Vakili, P. [1992] "Ordinal optimization of DEDS," *J. Discrete Event Dynam. Syst.* **2**(2), 61–88.

Ho, Y. C., Zhao, Q. C. and Jia, Q. S. [2007] *Ordinal Optimization: Soft Optimization for Hard Problems* (Springer).

Holland, J. H. [1975] *Adaptation in Natural and Artificial Systems* (The University of Michigan Press).

Homem-de-Mello, T., Shapiro, A. and Spearman, M. L. [1999] "Finding optimal material release times using simulation-based optimization," *Manag. Sci.* **45**, 86–102.

Hsieh, B. W., Chen, C. H. and Chang, S. C. [2001] "Scheduling semiconductor wafer fabrication by using ordinal optimization-based simulation," *IEEE Trans. on Robot. Automat.* **17**(5), 599–608.

Inoue, K. and Chick, S. E. [1998] "Comparison of Bayesian and Frequentist assessments of uncertainty for selecting the best system," *Proc. 1998 Winter Simulation Conf.*, pp. 727–734.

Inoue, K., Chick, S. E. and Chen, C. H. [1999] "An empirical evaluation of several methods to select the best system," *ACM Trans. on Model. Comput. Simulat.* **9**(4), 381–407.

Kim, S. H. and Nelson, B. L. [2003] "Selecting the best system: Theory and methods," *Proc. 2003 Winter Simulation Conf.*, pp. 101–112.

Kim, S.-H. and Nelson, B. L. [2006] "Selecting the best system," in *Handbooks in Operations Research and Management Science: Simulation*, eds. Henderson, S. G. and Nelson, B. L. (Elsevier), Chapter 18.

Kim, S. H. and Nelson, B. L. [2007] "Recent advances in ranking and selection," *Proc. 2007 Winter Simulation Conf.*, pp. 162–172.

Law, A. M. and Kelton, W. D. [2000] *Simulation Modeling and Analysis*, 3rd ed. (MacGraw-Hill).

Lee, L. H., Chew, E. P., Teng, S. Y. and Goldsman, D. [2004] "Optimal computing budget allocation for multi-objective simulation models," *Proc. 2004 Winter Simulation Conf.*, pp. 586–594.

Lee, L. H., Chew, E. P. and Teng, S. Y. [2007] "Finding the Pareto set for multi-objective simulation models by minimization of expected opportunity cost," *Proc. 2007 Winter Simulation Conf.*, pp. 513–521.

Lee, L. H., Chew, E. P., Teng, S. Y. and Chen, Y. K. [2008] "Multi-objective simulation-based evolutionary algorithm for an aircraft spare parts allocation problem", *Eur. J. Oper. Res.*, **189**(2), 476–491.

Lee, L. H., Chew, E. P., Teng, S. Y. and Goldsman, D. [2010] "Finding the Pareto set for multi-objective simulation models," *IIE Tran.*, in press.

Lehman, E. L. [1996] "Same concepts of dependence," *Ann. Math. Stat.* **37**(5), 1137–1153.

Kleijnen, J. P. C. [2008] *Design and Analysis of Simulation Experiments* (Springer, New York).

Koenig, L. W. and Law, A. M. [1985] "A procedure for selecting a subset of size m containing the l best of k independent normal populations," *Communication in Statistics — Simulation and Communication* **14**, 719–734.

Kushner, H. J. and Yin, G. G. [2003] *Stochastic Approximation Algorithms and Applications*, 2nd ed. (Springer-Verlag).

L'Ecuyer, P. [1994] "Efficiency improvement and variance reduction," *Proc. 26th Conf. on Winter Simulation*, pp. 122–132.

L'Ecuyer, P., Demers, V. and Tuffin, B. [2006] "Splitting for rare-event simulation," *Proc. 2006 Winter Simulation Conf.*, pp. 137–148.

Luo, Y. C., Guignard-Spielberg, M. and Chen, C. H. [2001] "A hybrid approach for integer programming combining genetic algorithms, linear programming and ordinal optimization," *J. Intell. Manuf.* **12**(5–6), 509–519.

Matejcik, F. J. and Nelson, B. L. [1995] "Two-stage multiple comparisons with the best for computer simulation," *Oper. Res.* **43**, 633–640.

Nelson, B. L. [1990] "Control-variates remedies," *Oper. Res.* **38**.

Nelson, B. L., Swann, J., Goldsman, D. and Song, W. M. [2001] "Simple procedures for selecting the best simulated system when the number of alternatives is large," *Oper. Res.* **49**, 950–963.

Pflug, G. C. [1996] *Optimization of Stochastic Models* (Kluwer Academic).

Pritsker, A. [1986] *Introduction to simulation and SLAM II* (John Wiley and Sons, New York).

Pujowidianto, N. A., Lee, L. H., Chen, C. H. and Yep, C. M. [2009] "Optimal computing budget allocation for constrained optimization," *Proc. 2009 Winter Simulation Conf.*, Austin, TX.

Rinott, Y. [1978] "On two-stage selection procedures and related probability inequalities," *Communications in Statistics* **7**, 799–811.

Rubinstein, R. Y. and Shapiro, A. [1993] *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method* (John Wiley & Sons).

Sanchez, S. M. [2005] "Work smarter, not harder: Guidelines for designing simulation experiments," *Proc. 2005 Winter Simulation Conf.*, pp. 178–187.

Shortle, J. and Chen, C. H. [2008] "A preliminary study of optimal splitting for rare-event simulation," *Proc. 2008 Winter Simulation Conf.*, Miami, FL, pp. 266–272.

Shi, L. and Chen, C. H. [2000] "A new algorithm for stochastic discrete resource allocation optimization," *J. Discrete Event Dynam. Syst. Theor. Appl.* **10**, 271–294.

Shi, L. and Olafsson, S. [2000] "Nested partitions method for global optimization," *Oper. Res.* **48**, 390–407.

Shi, L. and Olafsson, S. [2008] *Nested Partitions Optimization: Methodology and Applications* (Springer, New York).

Spall, J. C. [1992] "Multivariate stochastic approximation using simultaneous perturbation gradient approximation," *IEEE Trans. on Automat. Contr.* **37**, 332–341.

Sullivan, D. W. and Wilson, J. R. [1989] "Restricted subset selection procedures for simulation," *Oper. Res.* **37**, 52–71.

Swisher, J. R., Jacobson, S. H. and Yücesan, E. [2003] "Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey," *ACM Trans. Model. Comput. Simulat.* **13**(2), 134–154.

Trailovic, L. and Pao, L. Y. [2004] "Computing budget allocation for efficient ranking and selection of variances with application to target tracking algorithms," *IEEE Trans. Automat. Contr.* **49**, 58–67.

Wan, H., Ankenman, B. and Nelson, B. L. [2003] "Controlled sequential bifurcation: a new factor-screening method for discrete-event simulation," *Proc. 2003 Winter Simulation Conf.*, pp. 565–573.

Wilson, J. R. [1993] "Antithetic sampling with multivariate inputs," *Am. J. Math. Manag. Sci.* **3**, 121–144.

Zhao, Q. C., Ho, Y. C. and Jia, Q. S. [2005] "Vector ordinal optimization," *J. Optim. Theor. Appl.* **125**(2), 259–274.

Index