OPTIMIZATION OF RESOURCE ALLOCATION AND LEVELING USING GENETIC ALGORITHMS

By Tarek Hegazy,[1] Member, ASCE

ABSTRACT: Resource allocation and leveling are among the top challenges in project management. Due to the complexity of projects, resource allocation and leveling have been dealt with as two distinct subproblems solved mainly using heuristic procedures that cannot guarantee optimum solutions. In this paper, improvements are proposed to resource allocation and leveling heuristics, and the Genetic Algorithms (GAs) technique is used to search for near-optimum solution, considering both aspects simultaneously. In the improved heuristics, random priorities are introduced into selected tasks and their impact on the schedule is monitored. The GA procedure then searches for an optimum set of tasks' priorities that produces shorter project duration and better-leveled resource profiles. One major advantage of the procedure is its simple applicability within commercial project management software systems to improve their performance. With a widely used system as an example, a macro program is written to automate the GA procedure. A case study is presented and several experiments conducted to demonstrate the multiobjective benefit of the procedure and outline future extensions.

INTRODUCTION

Few companies can remain competitive in today's highly competitive business environment without effectively managing the cost of resources. In practice, basic PERT and CPM scheduling techniques have proven to be helpful only when the project deadline is not fixed and the resources are not constrained by either availability or time. Since this is not practical even for small-sized projects, several techniques have been used to modify CPM results in account of practical considerations. In dealing with project resources, two main types of techniques have been used: resource allocation and resource leveling. Resource allocation (sometimes referred to as constrained-resource scheduling) attempts to reschedule the project tasks so that a limited number of resources can be efficiently utilized while keeping the unavoidable extension of the project to a minimum. Resource leveling (often referred to as resource smoothing), on the other hand, attempts to reduce the sharp variations among the peaks and valleys in the resource demand histogram while maintaining the original project duration (Moselhi and Lorterapong 1993). These techniques, as such, deal with two distinct subproblems that can only be applied to a project one after the other rather than simultaneously. Accordingly, they do not guarantee (either individually or combined) a project schedule that minimizes the overall project time or cost (Karshenas and Haber 1990).

In this paper, an attempt is made to develop a practical procedure that searches for a near-optimum solution to resource allocation and leveling, simultaneously. The paper starts with a brief description of the advantages and limitations of current optimization-based and heuristic approaches. Individual improvements to existing heuristics are then proposed and tested on a case study. A multiobjective optimization using the genetic algorithms (GA) technique is then described and coded in a macro program. The performance of the proposed GA procedure is then evaluated on the case study, and recommendations made.

RESOURCE ALLOCATION AND LEVELING HEURISTICS

Limited-resource allocation algorithms deal with a difficult problem that mathematicians refer to as a ''large combinatorial problem.'' The objective is to find the schedule duration that is shortest, as well as consistent with specified resource limits. There exist optimization methods as well as heuristic methods for solving the resource allocation problem that go back in time to the 1960s (e.g., Wiest 1964). Various approaches have been formulated to solve the problem optimally, including Integer Programming, branch-and-bound, and Dynamic Programming (Gavish and Pirkul 1991). None of these, however, is computationally tractable for any real-life problem size, rendering them impractical (Moselhi and Lorterapong 1993; Allam 1988).

Alternatively, heuristic approaches have been proposed for solving the resource allocation problem. These approaches apply selected heuristic (rules) that are based on activity characteristics, such as the ''minimum total-slack'' rule, to prioritize the activities that compete for the limited resource. Accordingly, the resource is given to the top-ranked activities and the others are delayed. When ties occur during the implementation of a rule (e.g., when two or more activities have the same total slack), another rule such as ''shortest duration'' can be used to break the tie. The scheduling process, as such, starts from the project's start time, identifying eligible activities according to the network logic and resolving the over-requirements of resources using the selected set of heuristic rules. The process, as such, ensures that all project activities are scheduled without violating the logical relationships or the resource constraints. However, this comes on the expense of the total project duration, which often exceeds the duration determined by the original CPM analysis.

Heuristic rules have the advantage of being simple to understand, easy to apply, and very inexpensive to use in computer programs. They are able to rationalize the scheduling process and make it manageable for practical-size projects (Talbot and Patterson 1979). Furthermore, research has identified rules such as the ''least total-slack'' and the ''earliest late-start,'' which generally provide good solutions (Davis and Patterson 1975). Almost all commercial software for planning and scheduling, therefore, utilizes heuristic rules to provide resource allocation capabilities. Despite these benefits, however, heuristic rules perform with varying effectiveness when used on different networks, and there are no hard guidelines that help in selecting the best heuristic rule to use for a given network. They, as such, cannot guarantee optimum solutions. Furthermore, their drawbacks have contributed to large inconsistencies among the resource-constrained capabilities of commercial project management software, as reported in recent surveys (Hegazy and El-Zamzamy 1998; Johnson 1992).

Resource-leveling algorithms, on the other hand, attempt to reduce peak requirements and smooth out period-to-period

[1]Asst. Prof. of Constr. Mgmt., Dept. of Civ. Engrg., Univ. of Waterloo, Waterloo, ON, Canada N2L 3G1. E-mail: tarek@uwaterloo.ca

Note. Discussion open until November 1, 1999. To extend the closing date one month, a written request must be filed with the ASCE Manager of Journals. The manuscript for this paper was submitted for review and possible publication on January 8, 1999. This paper is part of the ***Journal of Construction Engineering and Management***, Vol. 125, No. 3, May/June, 1999. ©ASCE, ISSN 0733-9634/99/0003-0167–0175/ $8.00 + $.50 per page. Paper No. 17338.

fluctuations in resource assignment without changing project duration. Typical resources considered include a rented piece of equipment that needs to be returned early or a number of skilled workers who need to be hired for the job. Optimal solutions for the resource-leveling problem are based on mixed integer program formulations (Shah et al. 1993; Easa 1989). Such formulations are NP-complete and optimal solutions are reached for small-sized construction projects only. Heuristic algorithms are therefore needed.

A well-known heuristic algorithm is the minimum moment algorithm (Harris 1978). The objective in this algorithm is to minimize daily fluctuations in resource use while keeping the total project duration unchanged. As a proxy to this objective, the algorithm minimizes the moment of the resource histogram around the horizontal axis (time, calculations presented later in more detail). To accomplish this objective, the algorithm starts from an early start schedule and shifts noncritical activities within their float times so as to cause no project delay. At each time step, the shift(s) that yields the maximum reduction in the histogram moment is selected. Despite the simple nature of resource-leveling heuristics and their wide implementation on commercial project management software, they can only produce good feasible solutions and by no means guarantee an optimum solution.

IMPROVING RESOURCE-ALLOCATION HEURISTICS USING BIASED PRIORITIES

Since it is not possible to select an optimum heuristic rule for a given project network, one common procedure is to try a series of heuristic rules and then select the schedule with minimum duration. This procedure, however, has little diversity since the number of effective rules to enumerate is small and it is not expected that less effective rules will change much when effective rules are not improving the schedule. Therefore, without introducing new rules or changing the mechanics of heuristic procedures, a simple approach of forcing random activity priorities is presented to improve the goodness of the schedule. The concept is demonstrated on a case study of a project with twenty activities and six resources. The case study data including activities' resource requirements and daily resource limits is presented in Table 1. This data was input to Microsoft (MS) Project software (Microsoft Project 1995) for quick analysis.

TABLE 1. Case Study Data

Activity (1)	Duration (days) (2)	Predecessors (3)	Resource Requirements per Day R1 (4)	R2 (5)	R3 (6)	R4 (7)	R5 (8)	R6 (9)
A	6	—	5	2	2	2	7	4
B	3	—	3	5	2	3	9	6
C	4	A	2	4	4	2	3	1
D	6	—	5	4	3	5	5	4
E	7	A, B	3	5	2	3	8	0
F	5	C	4	1	4	9	2	5
G	2	D	4	1	4	3	9	8
H	2	A, B	5	5	4	0	9	1
I	2	G, H	3	2	4	3	4	2
J	6	F	1	5	4	6	7	3
K	1	C, E	3	3	2	4	5	1
L	2	E, G, H	3	2	2	8	3	4
M	4	I, K	2	2	2	2	4	8
N	2	F, L	1	4	4	3	4	1
O	3	L	5	5	4	6	2	3
P	5	J, M, N	3	2	3	4	7	8
Q	8	O	4	5	4	2	3	4
R	2	D, O	5	3	3	3	7	8
S	6	P, R	2	4	6	2	3	4
T	2	Q	1	6	2	7	5	2
Daily Resource Limits			7	10	10	16	18	13

Without considering the given resource constraints, the total project duration, determined by simple CPM analysis, is 32 days. When the resource-leveling feature (leveling is used in the software's terminology for both allocation and leveling) of MS Project was set to ''Automatic,'' total project duration was extended to 49 days, avoiding resource over-allocations. This solution was obtained using the software's ''standard'' set of heuristic rules, which maintains logical relationships and applies the ''minimum total slack'' rule to resolve conflicts. The same results were also obtained using the ''minimum total slack'' rule on Primavera Software (Primavera 1995) as a high-end system. Several other heuristic rules were also tried on Primavera software, without improving the schedule. A project duration of 49 days is, therefore, the best result that can be obtained from widely used commercial software. It is noted that this result is obtained when all project activities have the same priority level.

Most commercial scheduling software systems allow users to specify priority levels to activities. MS Project implements that in a direct manner by allowing users to select among eight priority levels (''Highest,'' ''High,'' etc., to ''Lowest''), and assign it in a simple spreadsheet form. The software also provides a second set of heuristic rules for resource allocation in which activity priority takes precedence over its ''standard'' set of heuristic rules. It is possible, therefore, to introduce some bias into some activities and consequently monitor the impact on the schedule. As an example, consider the case when only activity (R) in the present case study is given ''Highest'' priority while all others are set to ''Lowest.'' With this limited change to the original schedule, the project duration substantially decreased to 46 days (Fig. 1), one of the solutions for that particular example obtained by Talbot and Patterson (1979) using optimization. This simple approach is therefore proven to provide better results than existing heuristics.

Since it is not possible to readily identify, from a given network, which activities to assign higher priorities than others to improve the schedule, a simple iterative procedure may be used. A flow chart of such a procedure is presented in Fig. 2. It starts by initializing the scheduling software by setting its resource allocation feature to ''Automatic'' and defining a set of heuristic rules, ''activity priority'' being the leading one. Afterwards, each activity in the project is selected in turn, given ''highest'' priority over all others, and the consequent project duration is monitored. If the project duration decreases at any step in the process, corresponding activity priorities are saved and the process continues to improve the schedule further. It is also possible to automate this procedure by writing a simple macro on the scheduling software. Despite its perceived benefit, however, the main shortcoming of this procedure is its inability to identify an optimum set of activities' priorities that reduces project duration the most. This issue is dealt with later using the GA.

IMPROVING RESOURCE LEVELING HEURISTICS USING DOUBLE MOMENTS

In the course of optimizing resource allocation, the schedule repeatedly changes and along with it are the daily demands of resources. It is the objective of project managers, therefore, to optimize both the allocation and the leveling aspects of resources. As mentioned previously, the minimum moment algorithm has been used as a heuristic approach to calculate a measure of the fluctuations in daily resource demands. This is represented in Fig. 3(a), where Histogram 1 and Histogram 2 are two alternative resource histograms, both having a total area of 40 resource days (i.e., equal total resource demands).

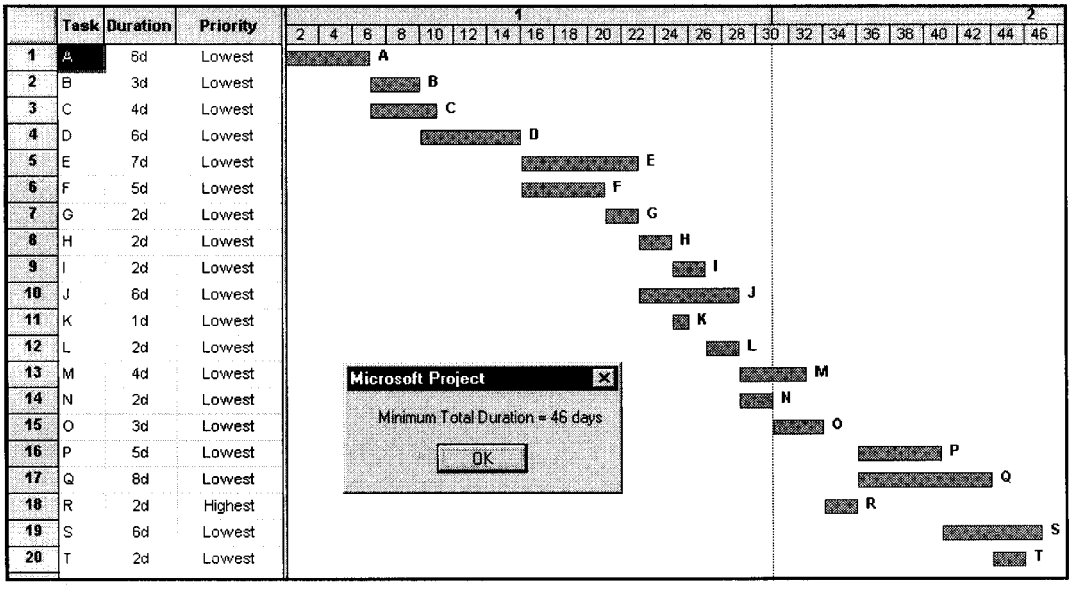

FIG. 1. Case Study Project with High Priority Assigned to Task (R)

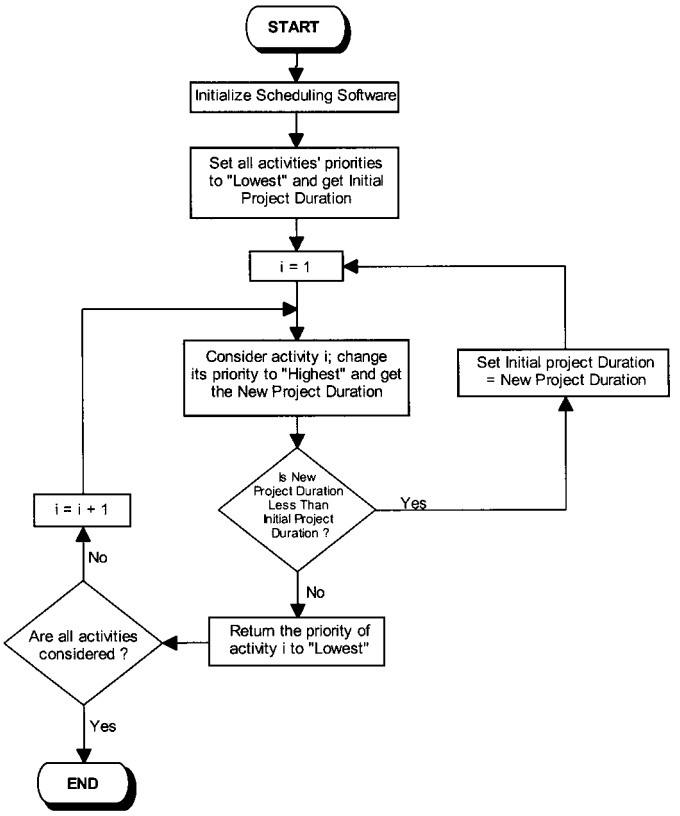

FIG. 2. Iterative Procedure for Improving Resource Allocation Heuristics

Histogram 1 is an ideal one, with a constant daily demand of four resource units, and no day-to-day resource fluctuations; the resource will be released after day 10. Histogram 2, on the other hand, exhibits high resource fluctuation with daily demand in the range of 2–6 resource units, and the resource will not be released until the end of day 12. The moment (M_x) of both histograms around the horizontal axis (days) are 160 and 166, respectively, representing a better resource leveling of Histogram 1. The moment M_x is calculated by summing the daily moments, as follows:

$$M_x = \sum_{j=1}^{n} \left[(1 \times \text{Resource Demand}_j) \times \frac{1}{2} \text{Resource Demand}_j \right] \quad (1)$$

where n = working-day number of the project's finish date. Or, for comparison reasons, (1) becomes

$$M_x = \sum_{j=1}^{n} (\text{Resource Demand}_j)^2 \quad (2)$$

While the minimum moment (M_x) method can be used to compare among histograms in terms of resource fluctuation, it does not take into consideration the resource utilization period. The latter is very important to minimize, particularly for equipment resources that are shared among projects or rented from external sources. Fig. 3(b), for example, shows a resource histogram having the same 40 resource days (total area), a maximum resource demand of 4, and a utilization period that extends till the end of day 13. Its M_x is 160, the same as that of Histogram 1, indicating resource fluctuation similar to Histogram 1 and better than Histogram 2, regardless of its 3- and 1-day extensions, respectively, beyond the two histograms. The single moment M_x, therefore, does not consider for the extended assignment of the resource. To overcome that, the moment M_y (around the vertical axis, resource amount) is computed as follows:

$$M_y = \sum_{j=1}^{n} [(1 \times \text{Resource Demand}_j) \times j] \quad (3)$$

Using (3), the (M_y) values calculated for the three resource histograms of Figs. 3(a and b) are 220, 255, and 316, respectively. The value of M_y, as such, gets higher as the resource remains employed in the project till a later date. Accordingly, M_y can be used as a good indicator of the resource release date in the project. Also, a simple modification to (3) can be used to calculate the moment M_y around a vertical axis that corresponds to the first day the resource is employed in the project [k, Fig. 3(c)]. In this case, the value of M_y represents the resource utilization period, irrespective of when the resource is employed or released, expressed as follows:

$$M_y = \sum_{j=k}^{n} [(1 \times \text{Resource Demand}_j) \times (j - k)] \quad (4)$$

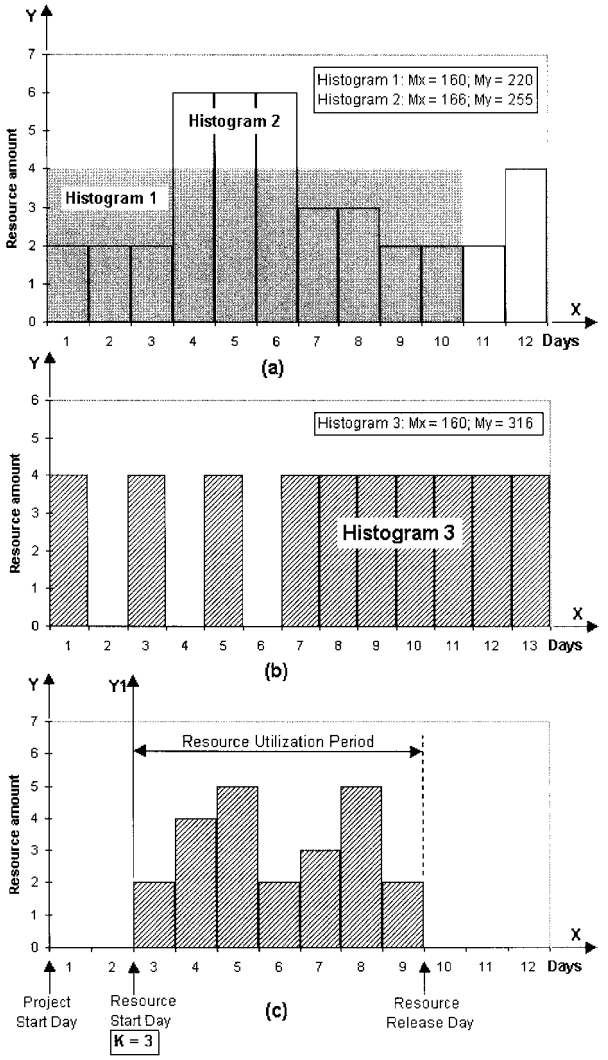

FIG. 3. Resource Histogram and Moment Calculations

Having the moment calculations defined, a project manager may use them as modified heuristics in four ways, according to his resource management objectives: (1) Minimize the M_x alone when the focus is on reducing daily resource fluctuations; (2) minimize the M_y of Eq. (4) alone when the focus is on reducing the resource utilization period; (3) minimize the M_y of Eq. (3) alone when the focus is on releasing the resource at an early date; or (4) minimize the double moments ($M_x + M_y$) when the focus in on both aspects. Incorporating such heuristics into a unified procedure for resource management is discussed in the next section.

MULTIOBJECTIVE OPTIMIZATION SEARCH USING GENETIC ALGORITHMS

Individual optimization of resource allocation or leveling has not been a simple task, let alone their simultaneous optimization. Given the modified heuristics presented in this paper, the objective can be restated, in a heuristic sense, as the search for a near-optimum set of activities' priorities that minimizes the total project duration under resource constraints while also minimizing the appropriate moment(s) of selected resources. This objective has a direct relationship to project cost minimization, which cannot be adequately achieved using mathematical optimization techniques. A schedule that efficiently employs limited resources, avoids daily fluctuation, and reduces project duration is eventually less costly. To deal with these multiobjectives, a search technique based on artificial intelligence, GAs, is used. Analogous to natural selection and genetics in reproduction, GAs have been successfully adopted to solve many science and engineering problems (Feng et al. 1997; Hegazy and Moselhi 1994). GAs also have been proven to be an efficient means for searching optimal solutions in a large problem domain such as the one at hand.

GAs are, in essence, optimization search procedures inspired by the biological systems' improved fitness through evolution. GAs employ a random-yet-directed search for locating the globally optimal solution. Typically, GAs require a representation scheme to encode feasible solutions to the optimization problem. Usually this is done in the form of a string called a chromosome (or gene). Each gene represents one member, i.e., one solution, that is better or worse than other members in a population. The fitness of each gene is determined by evaluating its performance with respect to an objective function. To simulate the natural ''survival of the fittest'' process, best genes exchange information to produce offspring that are evaluated in turn and can be retained only if they are more fit than the others in the population. Usually the process is continued for a large number of offspring generations until an optimum gene is arrived at.

Implementing the GA technique for the problem at hand involved five primary steps: (1) Setting the gene structure; (2) deciding the gene evaluation criteria (objective function); (3) generating an initial population of genes; (4) selecting an offspring generation mechanism; and (5) coding the procedure in a computer program. First, the gene structure was set as a string of elements, each corresponding to a priority level assigned to an activity, as shown in Fig. 4. As such, each gene represents one possible solution to the problem. To evaluate genes, an objective function can be constructed by eliciting the user's preference (or weights) among the multiobjectives. For example, assume a project with (r) resources, initial project duration D_0 determined by any resource allocation heuristic rule, initial M_x moment of every (j) resource (M_{xj0}), and initial M_y moment of every (j) resource (M_{yj0}). The values D_0, M_{xj0}'s, and M_{yj0}'s are therefore constants associated with the best solution provided by the scheduling software, before the GA procedure is applied. The user then needs to input the weight W_d of his preference in minimizing project duration and the weights W_j's of his preference in leveling every resource (j). In addition, the user needs to input the type of leveling moment (i.e., M_x, M_y, or $M_x + M_y$) that needs to be minimized for every resource (j). The weights and moment types are also constants representing the project manager's objective.

When a gene (i) is being evaluated, its priority values are assigned to the project activities to produce a new schedule

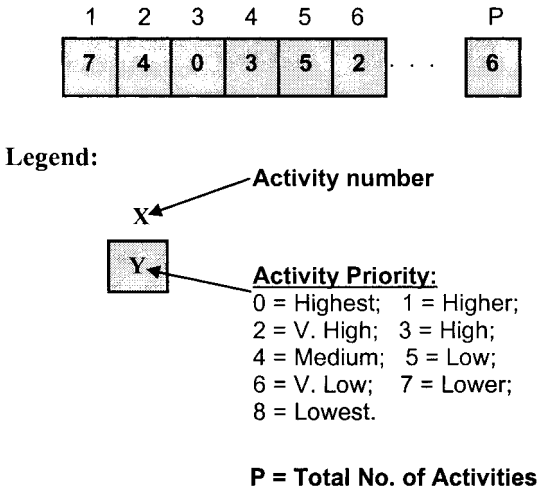

FIG. 4. Gene Formation

with duration D_i, in addition to new moments M_{xji} and M_{yji} for every resource (j). The fitness of that schedule (i.e., the fitness of its gene) is then determined by the relative improvement it exhibits over the initial schedule, as computed by an objective function that has two components for duration and moments, as follows:

$$W_d \cdot (D_i/D_0) + \sum_{j=1}^{r} [W_j \cdot (M_{xji} + M_{yji})/(M_{xj0} + M_{yj0})] \qquad (5)$$

The smaller this fitness value below 1.0, the less the duration and/or applicable comments, and accordingly the more fit the gene is. It is noted that the objective functions of (5) consider the minimization of both resource fluctuation and utilization period (M_{xj}'s + M_{yj}'s) of all resources. If, however, the objective is to minimize only one aspect (e.g., M_{xj}) for any resource (j), the resource's M_{yj} component in the equation can be preset to zero, rather than calculated.

Once the gene structure and fitness function are set, GA's evolutionary optimization takes place on a population of parent genes. The simplest way to generate that population is randomly, if no information is available on any activity that must have a fixed priority level. Population size (number of genes) is also an important factor affecting the solution and the processing time it consumes. Larger population size (on the order of hundreds) increases the likelihood of obtaining a global optimum; however, it substantially increases processing time. In the present application the user is given the flexibility to input the population size. Once the population is generated, the fitness of each gene in this population is evaluated using the objective function (5), and accordingly its relative merit is calculated as the gene's fitness divided by the total fitness of all genes.

The reproduction process among the population members takes place by either crossover or mutation, resembling natural evolution. Crossover (marriage) is by far a more common process and can be conducted by selecting two parent genes, exchanging their information, and producing an offspring. Each of the two parent genes is randomly selected in a manner such that its probability of being selected is proportional to its relative merit. This ensures that the best genes have a higher likelihood of being selected, without violating the diversity of the random process. Also, the exchange of information between the two parent genes is done through a random process (Fig. 5). As opposed to crossover, which resembles the main natural method of reproduction (Goldberg 1989), mutation is a rare process that resembles the process of the sudden generation of an odd offspring that turns to be a genius. This can be done by randomly selecting one gene from the population and then arbitrarily changing some of its information. The benefit of the mutation process is that it can break any stagnation in the evolutionary process, avoiding local minimums.

Once an offspring is generated by either method, it is evaluated in turn and can be retained only if its fitness is higher than others in the population. Usually the process is continued for a large number of offspring generations until an optimum gene is arrived at. In the present application, the user is given the flexibility to input the number of offspring generations.

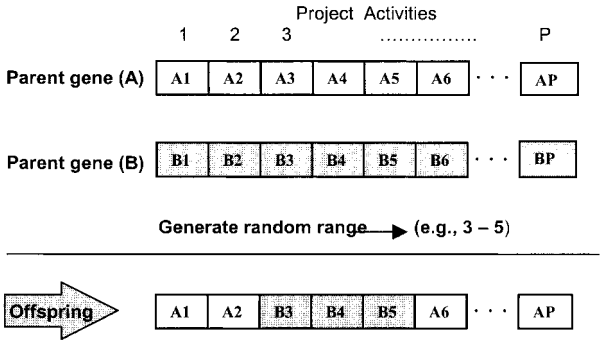

FIG. 5. Crossover Operation to Generate Offspring

PROCEDURE AUTOMATION AND EXAMPLE APPLICATION

Implementing the proposed GA procedure on commercial scheduling software simplifies the implementation process and provides project managers with an automated tool to improve the results of their familiar software. In this study Microsoft Project software is selected for implementing the GA procedure, for the reasons mentioned earlier as well as its ease of use and programmability features. The detailed GA procedure is outlined in Fig. 6. Using the macro language of Microsoft Project, the procedure was coded and then used to search for an optimum schedule for the case study at hand.

For simplicity, only one resource (R4) of the six resources in the present case study is assumed to be critical. As discussed previously, the software's initial solution to the resource-constrained schedule used ''Lowest'' priority for the project's 20 activities (column 2 of Table 2), producing a schedule of 49 days, in addition to M_x of 2,409, and M_y of 7,231 for resource R4. The GA optimization-search procedure was used to conduct four experiments with different objectives, as outlined in the second and third rows of Table 2. After initial experimentation with different population sizes and number of offsprings, a population of 200 genes and offspring of 1,000 was found to be a reasonable compromise between diversity and processing time for this size of problem. Accordingly, these were fixed for all experiments. Also, to avoid stagnation, crossover operation was set to be responsible for 95% of offspring generations while mutations was set to only 5%. Once the procedure was activated, an input screen, shown in Fig. 7 for Experiment 2, was displayed, requesting user input regarding GA parameters and the weights needed to formulate the objective function. The GA procedure then performed the optimization search, producing an output screen as shown in Fig. 8. The activity priorities resulting from the four experiments are shown in Table 2 along with the associated project durations and moment calculations.

It can be seen from the results of Table 2 that each experiment improved the schedule in a manner that is consistent with its objective. In Experiment 1 the objective was to solely minimize project duration, and accordingly a 44-day schedule was obtained (Fig. 9). This is 5 days shorter than the initial schedule and is also 2 days less than the 46-day schedule of the iterative process discussed earlier. Giving a 50% weight to minimizing the M_x of R4, Experiment 2 produced the smallest resource fluctuation moment of all the experiments (2,265). This was also reflected on the daily fluctuation range of R4 demand, which decreased from the initial 12 units to 10 units (Table 2). This experiment also decreased project duration to 45 days. Experiment 3 attempted to equally minimize the resource utilization moment (M_y) and the project duration, resulting in best improvements to both. Experiment 4 also attempted to minimize the three aspects of project duration, R4 fluctuation, and R4 utilization period, resulting in improvements to the three. Based upon these results, the case study clearly shows the benefits of the GA procedure in optimizing both resource allocation and leveling to improve scheduling results over those of existing heuristic procedures and commercial scheduling software systems. Clearly these benefits, in terms of shorter duration and better resource utilization, can be readily translated into cost savings as a function of indirect cost, incentive gains, and reduced resource rental or salary amounts.

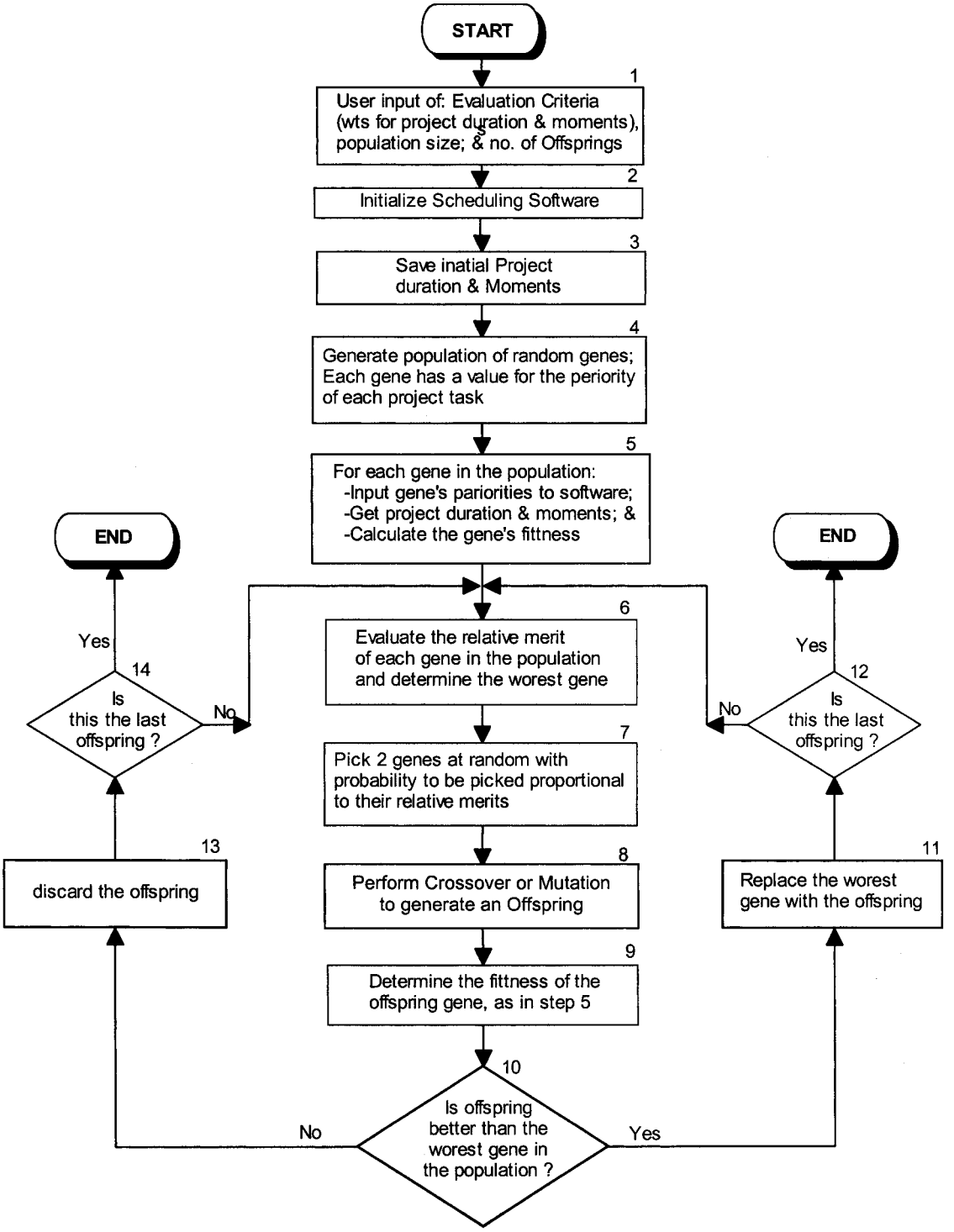

FIG. 6. Genetic Algorithms Procedure

In terms of processing time, the four GA experiments for the present 20-activity network took from 50 to 120 min on a Pentium 233 MMX PC to complete 1,000 offspring generations. To further examine the performance of the GA procedure on larger networks, several other experiments were conducted on networks with 40, 100, and 200 activities. Each of these networks was constructed by copying the 20 activities of the present case study several times. The GA procedure was then applied to each network for varying number of offspring generations 50, 100, and 200. While the GA procedure performed consistently and improved all schedules achieving an average of 10% reduction in duration, larger networks exhibited a noticeable increase in processing time (Fig. 10). Interestingly, however, larger networks arrived at best solutions earlier in the process than smaller ones, even with a lesser number of offspring generations (50 and 100). As such, the number of offsprings could be substantially reduced to decrease processing time. For practical-size projects with hundreds of activities, a population of 100 genes and intervals of 100 offspring could be used. Also, an overnight run may be a good option, given the potential saving in duration and resource-oriented savings. For interested readers, the GA procedure and the case study can be downloaded from the writer's web site, www.civil.uwaterloo.ca/tarek, under My Free Educational Software.

TABLE 2. Results of Genetic Algorithm Experiments

(1)	Initial Schedule (2)	Experiment 1 (3)	Experiment 2 (4)	Experiment 3 (5)	Experiment 4 (6)
GA Optimization Objective(s)	None	Min. Project Duration	Min. Duration + Min. Daily Fluctuation of R4	Min. Duration + Min. Utilization Period of R4	Min. Duration + Min Fluctuation of R4 + Min. Utilization Period of R4
Gene Evaluation Criteria & Weights	None	---------- Proj. Dur. 100%	Proj. Dur. 50% Mx of R4 50%	Proj. Dur. 50% My of R4 50%	Proj. Dur. 50% Mx+My of R4 50%
Activity		**Activity Priority**		**Results**	
A	Lowest	Highest	Very High	Higher	Higher
B	Lowest	Medium	Very Low	Low	Medium
C	Lowest	Very High	Medium	Higher	High
D	Lowest	Very High	Very High	Medium	Higher
E	Lowest	Very High	Lowest	Higher	Highest
F	Lowest	Very High	High	Medium	Low
G	Lowest	Medium	Lowest	Low	Low
H	Lowest	Lowest	High	Lowest	Very Low
I	Lowest	Very High	High	Very Low	Very Low
J	Lowest	Medium	High	Very Low	Lower
K	Lowest	Medium	Higher	Medium	Higher
L	Lowest	Medium	High	Lower	Very Low
M	Lowest	Very High	Very High	Medium	High
N	Lowest	Very Low	Medium	Low	Very High
O	Lowest	High	Very High	High	High
P	Lowest	Very High	Medium	Low	Lower
Q	Lowest	Lowest	Lowest	Lower	Low
R	Lowest	Higher	Very High	Low	High
S	Lowest	Medium	Low	Highest	Very High
T	Lowest	Lowest	Higher	Very High	Low
Project Duration	49	44	45	44	45
Calculated Moments of R4 Mx	2409	2381	2265	2375	2345
My	7231	6752	6952	6746	6832
Mx+My	9640	9133	9217	9121	9177
Range of R4	12	10	10	10	10
Utilization Period of R4	Day 1 to Day 49	Day 1 to Day 44	Day 1 to Day 45	Day 1 to Day 44	Day 1 to Day 45

Notations:
- Proj. Dur. = Project Duration.
- Mx = Moment around the x-axis (Time) of resource histogram
- My = Moment around the y-axis (Number of Resources) of resource histogram
- Range = (Maximum – Minimum) amount of the resource needed per day

COMMENTS ON ALGORITHM PERFORMANCE

The proposed GA procedure is, in essence, a heuristic search algorithm that attempts to optimize the schedule. It has been demonstrated to have several interesting characteristics, including the following:

- It attempts to improve on an existing schedule determined using all the power features of commercial project management software.
- The GA approach is an efficient search procedure that arrives at solutions by searching only a small fraction of the total search space. With 20 activities, each having 8 options for its priority, the total search space is 8^{20}. It took only 1,000 offspring (involving a search space of 20,000) to arrive at the results of Table 2.
- It combines both resource allocation and leveling into the objective function for the GA search.
- Since the GA procedure works on top of scheduling software, activities' cost data may not be available in this type of software. The formulation of the GA procedure and its objective function, therefore, have costs implied by the calculated moments, without requiring additional user input.
- The GA approach and its objective function can be modified to incorporate other objectives—for example, those related to selecting the appropriate methods of construction to use in the different tasks so that a certain deadline is met in the least-costly manner. This adds a time-cost trade-off dimension to the GA search. Such an extension can consider for the the daily penalty of exceeding the deadline and also for the incentive for early completion. As opposed to mathematical optimization, the GA procedure will work regardless of the complexity of the model. Implementation of these extensions is currently being investigated by the writer.

The main downside of the algorithm is its random nature, which requires a long processing time. One option is to code

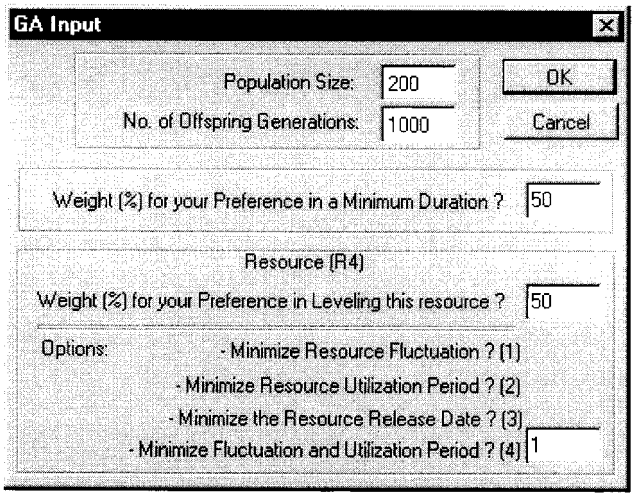

FIG. 7. GA Input Screen for Experiment 2

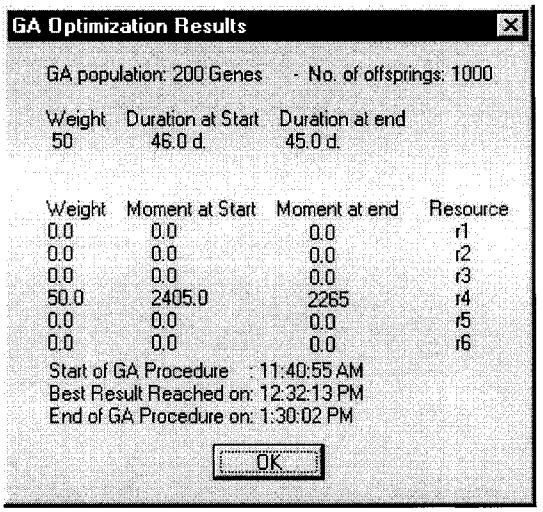

FIG. 8. GA Output Screen for Experiment 2

the procedure in a faster programming language than the VBA language included with Microsoft Project, one such as C or C++. Another option is to code it to work as a memory resident program that runs automatically, similar to screen savers

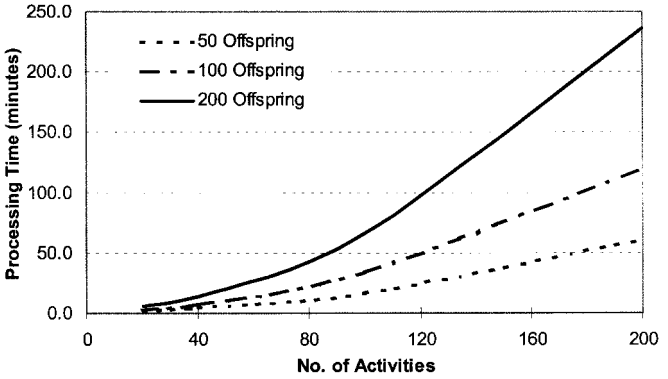

FIG. 10. Processing Time versus Number of Activities

when the process is set idle for some time. Accordingly, it can propose improvements to the schedule at times when sufficient processing is possible.

In addition to these programming improvements, there are several other ways to improve the performance of the GA procedure. Some changes could be made to the basic genetic algorithm formulation to make it faster and more efficient in similar types of problems. This is currently an area of extended research among researchers who perceive the many benefits of using this technique in many applications within the civil engineering domain. Also, since the heuristic rules underlying the GA procedure are constant in the optimization, it is possible to experiment with a different set of heuristic rules. The ''activity priority'' rule still has to be the leading one; however, it could be followed by any rule other than the ''minimum total slack,'' such as ''smallest duration.'' This may lead to further improvement in the schedule. Furthermore, one possible approach to speed the GA procedure is to combine into it the iterative procedure of Fig. 2, as a preprocessor. Step 2 of the GA procedure (Fig. 6) can therefore be changed to activate the iterative procedure. This approach tries to arrive at quick improvements to the schedule upon which the GA procedure works.

It is noted that the implementation of the GA procedure on Microsoft Project software benefited from the software's feature of allowing user-specified priorities to activities. Other software systems such as Primavera, for example, do not directly allow for that and as such requires some manipulation.

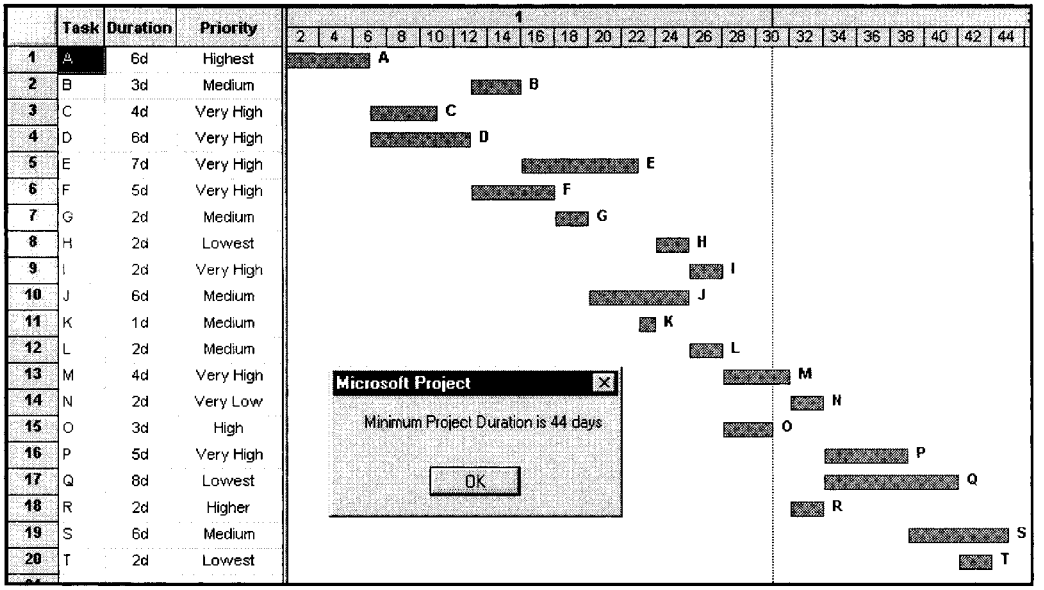

FIG. 9. Optimum Schedule of Experiment 1

The user, for example, can specify a custom activity code called ''priority,'' containing a number that represents the priority level of each activity. This code can then be used as the leading heuristic rule for resource allocation and for the implementation of the GA procedure.

SUMMARY AND CONCLUDING REMARKS

Three main developments were made in this paper with respect to improving the resource management of projects: (1) An effective improvement to resource allocation heuristics using random activity priorities; (2) a practical modification to resource leveling heuristics using a double-moment approach; and (3) a multiobjective optimization of both resource allocation and leveling using the genetic algorithms technique. Using a widely used project management software, a macro program was written to automate the GA procedure and a case study was used to demonstrate its benefits and future improvements and extensions.

In recent years, project management software systems have been improving continuously and recent versions have exhibited better interfaces, integrated planning and control features, and Internet capabilities. Yet, basic project management functions such as resource allocation, resource leveling, and time-cost trade-off analysis have been the least improved. Still, to some practitioners software systems provide merely powerful presentation capabilities and real savings can be achieved only by putting a hammer to a nail. It is hoped that practical implementations of new approaches such as genetic algorithms justify the effort spent in proper planning and scheduling as keys to effective project management and ultimately to actual savings in project time and cost.

APPENDIX. REFERENCES

Allam, S. I. G. (1998). ''Multi-project scheduling: A new categorization for heuristic scheduling rules in construction scheduling problems.'' *J. Constr. Mgmt. and Economics*, E&FN Spon, 6(2), 93–115.

Davis, E. W., and Patterson, J. H. (1975). ''A comparison of heuristic and optimum solutions in resource-constrained project scheduling.'' *Mgmt. Sci.*, 21(8), 944–955.

Easa, S. (1989). ''Resource leveling in construction by optimization.'' *J. Constr. Engrg. and Mgmt.*, ASCE, 115(2), 302–316.

Feng, C., Liu, L., and Burns, S. (1997). ''Using genetic algorithms to solve construction time-cost trade-off problems.'' *J. Comp. Civ. Engrg.*, ASCE, 11(3), 184–189.

Gavish, B., and Pirkul, H. (1991). ''Algorithms for multi-resource generalized assignment problem.'' *Mgmt. Sci.*, 37(6), 695–713.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, Reading, Mass.

Harris, R. (1978). *Resource and arrow networking techniques for construction*. Wiley, New York.

Hegazy, T., and El-Zamzamy, H. (1998). ''Project management software that meet the challenge.'' *Cost Engrg. J.*, 4(5), 25–33.

Hegazy, T., and Moselhi, O. (1994). ''Analogy-based solution to markup estimation problem.'' *J. Comp. Civ. Engrg.*, ASCE, 8(1), 72–87.

Johnson, R. (1992). ''Resource constrained scheduling capabilities of commercial project management software.'' *Proj. Mgmt. J.*, 22(4), 39–43.

Karshenas, S., and Haber, D. (1990). ''Economic optimization of construction project scheduling.'' *J. Constr. Mgmt. and Economics*, E&FN Spon, 8(2), 135–146.

Li, H., and Loing, P. (1997). ''Using improved genetic algorithms to facilitate time-cost optimization.'' *J. Constr. Engrg. and Mgmt.*, ASCE, 123(3), 233–237.

Microsoft Project. (1995). *Reference manual, ver. 4.1 for Windows 95*. Microsoft Corporation, Redmond, Wash.

Moselhi, A., and Lorterapong, P. (1993). ''Least impact algorithm for resource allocation.'' *Can. J. Civ. Engrg.*, CSCE, 20(2), 180–188.

Premavera. (1995). *Reference manual, ver. 1.0 for Windows*. Primavera, Bala Cynwyd, Pa.

Shah, K., Farid, F., and Baugh, J. (1993). ''Optimal resource leveling using integer-linear programming.'' *Proc., 5th Int. Conf. in Comp. in Civ. & Bldg. Engrg.*, ASCE, Reston, Va., 1, 501–508.

Talbot, F., and Patterson, J. (1979). ''Optimal methods for scheduling projects under resource constrains.'' *Proj. Mgmt. Quarterly*, Dec., 26–33.

Wiest, D. (1964). ''Some properties of schedules for large projects with limited resource.'' *Operations Res.*, 12, 395–416.