DEVELOPMENT OF A RESOURCE LEVELING MODEL USING OPTIMIZATION

Jin-Lee Kim [1] and Ralph D. Ellis [2]

[1] Ph.D. Candidate, Department of Civil and Coastal Engineering, University of Florida, Gainesville, FL, USA
[2] Associate Professor, Department of Civil and Coastal Engineering, University of Florida, Gainesville, FL, USA
Correspond to jin5176@ufl.edu

ABSTRACT: This paper presents a GA-based optimal algorithm for a resource leveling model that levels the resources of a set of non-critical activities experiencing conflicts simultaneously up to an assumed level of resource rates specified by the planner using a pair-wise comparison of the activities being considered. A parameter called the future float is adopted and applied as an indicator for assigning leveling priorities to the sets of activities experiencing conflicts. A construction project network example was worked out to demonstrate the performance of the proposed method. The histogram obtained using the algorithm proposed was shown to be the same as, or very close to that produced by the existing resource leveling method based on the least total float rule, which shifts non-critical activities individually.

Key words: Resource Leveling, Algorithms, Optimization Models, Project Management, Scheduling

1. INTRODUCTION

Resource leveling in construction engineering is an effective scheduling strategy only when the project duration is fixed without regard to limitations in the availability of resources. Resource leveling problems need to be solved accurately to reduce the maximum demands for a given resource on any given day. A schedule that was generated using the early start time of an activity generally tends to create conflicts by demanding large numbers of resources (for example, workers, equipment, etc.) on some days of the project. Project planners usually level resources to meet the physical limit of resources and to avoid the day-to-day fluctuations in resource demands.

The fluctuation of resources is undesirable for the contractor because it causes cost overrun and inefficient resource management problems. Therefore, the resource leveling aims to minimize fluctuations in the patterns of resource usage within the required duration. The assumption underlying this problem is that the availability of resources is not limited and the project duration is fixed. A histogram showing the variation in the resource profile is generally created as a result of solving resource leveling problems. This histogram becomes an important tool in project management to avoid the difficulties associated with the large variations in resource usage.

Depending on the type of constraints, resources and time, the scheduling technique can be divided into two categories, i.e., resource allocation and resource leveling. Often, two techniques have been used interchangeably in the literature. However, it is necessary to differentiate resource leveling from resource allocation. Resource allocation aims to minimize the project duration with resource constraints, while resource leveling is used to minimize fluctuations in the patterns of resource usage within the required duration. In this paper resource leveling, rather than resource allocation, will be used to describe the scheduling technique to reduce the variations among the valleys and peaks in the resource demand.

Unlimited resource leveling problems have been solved using either mathematical models or heuristic rules. The heuristic procedures first develop a resource histogram based on the early start time of activities derived from Critical Path Method (CPM) or Program Evaluation and Review Technique (PERT) techniques, and then shift non-critical activities according to some particular rules. The notable point is that current heuristic techniques for resource leveling shift a non-critical activity individually to meet the resource requirements in a sequential mode, starting with activities that have the least total float. Although the heuristic methods can handle very large projects, the solution they provide is not an optimum. There is still a need for improvement in the area of optimality and efficiency since these methods have proven to be inconsistent with regard to the quality of results produced on project networks.

Relative to the vast amount of research that has been conducted on heuristic procedures, optimal procedures have rarely been the focus of such extensive research. The main disadvantage of these models is that they are not suitable for large networks because of the combinatorial mathematics of the leveling problem. If a project has 10 non-critical activities, each having a float of 5 days, for instance, approximately 10 million (5^{10}) possible combinations exist. Because of this, little research has been done to solve the resource leveling problem using optimization techniques.

This paper presents a GA-based optimal algorithm for a resource leveling model that levels the resources of a set of non-critical activities experiencing conflicts simultaneously up to an assumed level of resource rates specified by the planner using a pair-wise comparison of the activities being considered. A parameter called the future float is adopted and applied as an indicator for assigning leveling priorities to the sets of activities being considered. A construction project network example was worked out to demonstrate the performance of the proposed method.

2. PREVIOUS STUDIES

Recently, some research has been conducted to solve the resource leveling problem in construction engineering using optimization techniques such as genetic algorithms (GAs). Chan et al. [1] proposed a resource leveling method along with the resource allocation that employed GAs. They set a single equation with the objective of minimizing the difference between resource availability and utilization. In the string representation of GAs, they used the concept of the current float to set the scheduling priority. The current float concept [2] was introduced to eliminate network recalculation, which is the main disadvantage of the total float concept in the CPM analysis. The model establishes the order of priority for the relevant activities based on their current floats and allocates resources to the activity that has the smallest current float under the availability of resources.

Hegazy [3] combined resource allocation and leveling using GAs. The research incorporated the concept of minimum total float for resource allocation, and minimum moment method for resource leveling, as the decision variable. Although an effective improvement was made in the combined model, it has been stated clearly that the method did not yield optimum solutions. Leu and Yang [4] proposed a GA-based multicriteria optimal model for construction scheduling. Researchers unified time-cost trade-off and resource allocation. They used activity duration obtained from the time-cost trade-off model as basic input data for the computation of minimum project duration under resource constraints. However, they mentioned that improvements in resource leveling needed to be made because resource conflicts still occurred. Senouci and Eldin [5] presented a model that performs resource leveling and resource-constrained scheduling simultaneously using the quadratic penalty function to transform the resource-scheduling problem into an unconstrained one. In the string representation of GAs, however, they used the activity duration and start time that causes resource conflicts.

As a result, no optimal procedures have proven to be computationally feasible for large, complex projects that can occur in practice. Therefore, it is necessary to seek a systematic way to avoid resource conflicts that occur due to the early start schedule. In doing so, it is important to develop a more efficient algorithm that searches optimal solutions for the resource leveling problems for large-sized project networks in a reasonable amount of time. A developed algorithm should consider both the availability of resources and precedence relationships. Since this research takes GAs as a methodology to develop an algorithm for the resource leveling problems, a brief description of genetic algorithms (GAs) is provided in the next section.

3. GENETIC ALGORITHMS (GA_S)

A new optimization technique, genetic algorithms (GAs), has emerged as a tool that is beneficial for construction applications. GAs perform a random search for the optimal solution to a problem by simulating natural-evolution and survival-of-the-fittest mechanisms. GAs have attracted considerable attention in a number of fields not only as a methodology for optimization, adaptation, and learning, but also as optimization techniques for solving discrete optimization problems or other hard optimization problems. GAs differ from conventional optimization and search procedures in the following four ways, as summarized by Goldberg [6]:

- GAs work with a coding of the parameter (solution) set, not the parameters, (solutions) themselves,
- GAs search from a population of solutions, not a single solution,
- GAs use objective function (fitness function) information, not derivatives or other auxiliary knowledge, and
- GAs use probabilistic transformation rules, not deterministic ones.

The main components of GAs are the population, the string, and the feature [6]. Population is a set of possible solutions to the problem. String is a possible solution to the problem. It has been referred to as the “chromosome” in some research. Feature is a part of a solution to a given problem. It has been referred to as the “gene” in some research. Usually, a feature is an independent variable in the problem.

GAs start with an initial population of individuals generated at random. Each individual in the population represents a potential solution to the problem under consideration. The individuals evolve through successive iterations, called generations. During each generation, each individual in the population is evaluated using some measure of fitness. Then, the population of the next generation is created through genetic operators. The procedure continues until the termination condition is satisfied.

Three main genetic operators in GAs are reproduction, crossover, and mutation. They are used to create the next generation [6]. Reproduction is a process in which individual strings are generated according to their objective function values. Strings with a higher value have a higher probability of contributing one or more offspring in the next generation. The easiest way to implement the reproduction operator is to create a roulette wheel where each current string in the population has a roulette wheel slot sized in proportion to its fitness value. After reproduction, crossover can proceed in two steps. First, features of the newly reproduced strings in the mating pool are mated at random. Second, each pair of strings undergoes crossover. Mutation is the occasional random alteration of the value of a string position. The mutation operator plays a secondary role in GAs. It is notable that the frequency of mutation to obtain good results in empirical genetic algorithm studies is on the order of one mutation per thousand position transfers. Mutation rates are similarly small in natural populations, leading to the conclusion that mutation is appropriately considered as a secondary mechanism of genetic algorithm adaptation. The three basic parameters of GAs are crossover probability, mutation probability, and population size. The most important component in the GA approach is to decide the fitness function and evaluation function. It is necessary to distinguish between the evaluation function and the fitness function used by GAs. The evaluation function, also called the objective function, provides a measure of performance with respect to a particular set of parameters. In contrast, the fitness function transforms that measure of performance into an allocation of reproductive opportunities. The evaluation of a string representing a set of parameters is independent of the evaluation of any other string. However, the fitness of that string is always defined with respect to other members of the current population [7].

4. MODEL DEVELOPMENT

4.1 Model Formulation

The unlimited resource leveling is described as "a process wherein the activities in the network are positioned in time such that the project resources are minimized on a day-to-day basis [8]." Depending on a critical path network, a model assumes the activities, the network logic, and resources for solving the unlimited resource leveling problems. The major assumptions are the following:

- Once an activity is started, it will be completed without interruption,
- The resource requirements of each activity are specified and constant during the duration of the activity,
- Duration assigned to each activity is constant during the period of the activity,
- Precedence relationship between any activities is maintained, and
- The project's completion date is fixed so that it cannot be extended or shortened.

It is notable to differentiate an optimal solution from a near optimal solution because both terms have very specific meanings in the operations research literature. An optimal algorithm is one that guarantees that an optimal solution will be found as soon as an optimal algorithm is completed. A near optimal algorithm, however, has a slightly less precise meaning than an optimal algorithm. Upon completion of a near optimal algorithm, a near optimal solution is usually produced. What is meant by a near optimal solution is that it is a solution close to the optimum.

4.2 Genetic Algorithm Procedure for Resource Leveling

The simplified procedure of GAs that is used in this paper is as follows: (1) Define a solution representation, (2) Set variables, objective functions, and constraints, (3) Generate initial population of solutions (strings), (4) Evaluate the fitness of possible solution, (5) Apply genetic operators, and (6) Test termination conditions.

Step 1: Defining a solution representation

A solution is represented by the set of values associated with the problem variable. The unlimited resource leveling problem needs to be transformed to an unconstrained problem since the original GA does not allow for any constraints but directly solve only unconstrained optimization problems. This can be accomplished by adding a penalty function to the genetic algorithm. Whenever a violation of any of the constraints occurs, the objective function is penalized.

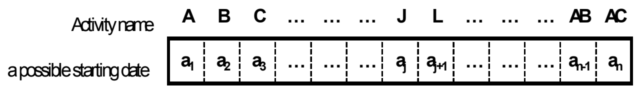
Activity name
A B C … … … J L … … … AB AC
a possible starting date
a_1 a_2 a_3 … … … a_j a_{j+1} … … … a_{n-1} a_n

Figure 1. Genetic algorithm representation of solution

Figure 1 shows a possible solution represented in a string. A gene value inside a box stands for the delay in the starting time of an activity. A detailed description of a gene value is given in Step 2.

Step 2: Setting the variables, objective functions, and constraints

Variables are the delay in the starting time of an activity (i.e., value inside each gene), which was first used by

Sathyanarayana et al. [9] to obtain optimum solutions for a resource allocation problem. A different concept of the delay computation method is used in the present model, which is derived from the concept of the future float. The future float (FF), of an activity, introduced by Moselhi and Lorterapong [10], is applied as an indicator for assigning leveling priorities to the sets of activities being considered. It is defined as

$$FF_i = LST_i - NTF$$
$$NTF = CT + \min (d_1, d_2, \ldots, d_j, d_{j+1}, \ldots, d_k)$$

where, i = the activity that will be postponed to the next time frame (NTF), j = the activity to be scheduled at CT, k = the total number of activities to be scheduled at CT, FF_i = the future float of activity i, LST_i = the latest start time of activity i from the original CPM analysis, and CT = current time [2]. An activity with a lower gene value is set ahead of others with higher gene values in the same rank.

The objective function used in this paper is to minimize the deviation between actual resource requirements and the desirable resource rates established by the scheduler for each time unit during the whole project duration, which is the typical pursuit of most algorithms in unlimited resource leveling problems [1] [3] [4]. The objective function can be formulated as:

$$\textit{\textbf{minimize}}\ \text{URLI},\ \Lambda = \sum_{i=1}^{T} Rd_i = \sum_{i=1}^{T} (Ra_i - \sum_{j=1}^{A} Rr_{ij}) + P$$

where, URLI, Λ = the unlimited resource leveling index, Rd_i = the difference between available and required resource on day i, Ra_i = the resource available on day i, Rr_{ij} = the resource required on day i by the jth activity, T = the fixed target project duration, $j \in A$ = set of all the activities scheduled on day i, and P = a penalty value whose purpose is to prevent the violation of precedence relationships. It is notable that the project duration should not exceed the fixed one as assumed earlier.

Step 3: Generating initial population of solutions (strings)

Since genetic algorithms start with the generation of an initial population of random solutions, which are strings or chromosomes, an initial population is created at random to apply GAs, as is done in most research [1] [3] [4] [11]. Population size indicates how many strings in a population are in one generation. It is one of the most important factors that affect the solution and computation time of genetic algorithms. Essentially, a large population size will increase the accuracy of obtaining a global optimal solution, but require large amount of computation time. If too few strings exist, GAs have few opportunities to perform crossover and only a small part of the search space is explored. In contrast, if too many strings exist, then GAs slow down. Goldberg [6] has shown that an increase in population size is not useful because it increases the length of the problem solving process. The size of the initial population is set at 30, the lowest acceptable population size as suggested in the previously conducted studies [6] [12].

Step 4: Evaluating the fitness of possible solutions

The fitness of each string in the population obtained in Step 3 is evaluated by the value calculated when the fitness of string i is divided by the sum of the total fitness of all strings [7].

Step 5: Applying genetic operators

First, reproduction measures the fitness of strings in a generation. Some of the strings are reproduced in proportion to their objective function values. The aim of reproduction is to provide good solutions with a higher chance of passing their feature to the next generation than bad ones. Reproduction also increases or decreases the number of offspring for each string in the population according to the fitness values.

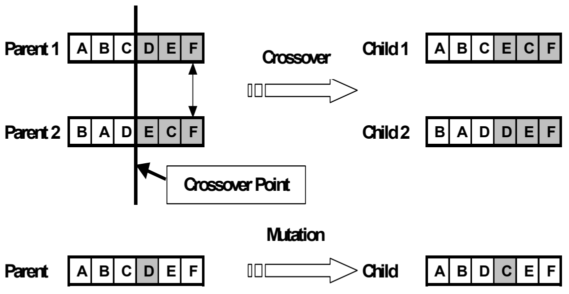

Figure 2. Genetic operators – Crossover and Mutation

Second, crossover selects two distinct strings from the population at random. It exchanges some portion of the strings with a probability equal to the crossover rate in order to create a new offspring. One of the simplest ways is to choose some crossover point randomly. Everything before this point is copied from a first parent and everything after a crossover point is copied from a second parent (See Figure 2). Crossover probability indicates how often crossover will be performed. If no crossover takes place, an offspring is an exact copy of the parents. Otherwise, an offspring is created from part of the parents' string. If crossover probability is 0%, a whole new generation is created from exact copies of strings from the old population, even though the new generation is not exactly same. If 100%, then all offspring are created by crossover.

Finally, uniform mutation, which is one type of mutation, takes place after a crossover is performed. Mutation is a random change of features in a string to reintroduce lost bit values into a population. It alters one or more features of a selected string with a probability equal to the mutation rate. Mutation changes the new offspring at random (See Figure 2). Mutation probability indicates how often parts of a string will be mutated. If no mutation occurs, an offspring is created after crossover without any change. If mutation is performed, part of a string is changed. If mutation probability is 0%, then nothing is changed. However, if 100%, then the whole string is changed. Mutation probability is necessary to prevent a local optimum of a solved problem, in which all solutions in the population fall into an optimum.

Step 6: Testing termination conditions

An algorithm for the resource leveling model is terminated when it meets conditions. The genetic algorithm procedure is terminated if a resource histogram is close to a rectangle within the fixed project duration. Because the nearer the value of the unlimited resource leveling index (URLI) is to zero, the more closely the resource histogram resembles a rectangle.

4.3 Case Example

This research team chose a 29-activity network that was extracted from Willis and Hastings [13] to illustrate a problem solved by the genetic algorithms procedure proposed in this paper. Activities of a project example with its duration and resources required for each activity is tabulated in Table 1.

Table 1. Detail information of activities (After [13])

Activity name	Dependency	Duration (Weeks)	Number of Resource requirement per week					
			Labors	Steel workers	Concrete workers	Brick layers	Cranes	Dump trucks
10	-	5	2	1	-	-	-	-
20	10	4	4	-	2	-	1	1
30	20	3	2	-	-	-	-	-
40	30	4	3	1	-	-	1	-
50	40	3	-	2	-	-	-	-
60	50, 280	1	-	2	-	-	-	-
70	50, 280	3	2	-	-	2	-	-
80	60, 70	1	1	1	-	-	-	-
90	50, 280	6	2	1	-	2	-	-
100	90	2	2	3	-	-	-	-
110	-	8	6	-	-	-	-	1
120	20, 110	4	4	-	-	-	-	1
130	120	12	2	-	4	-	-	-
140	130	1	2	2	-	-	-	-
150	140, 190	1	2	2	-	-	-	-
160	150	-	-	-	-	-	-	-
170	120	4	1	-	-	2	-	-
180	40, 170	1	2	1	-	-	1	-
190	180	1	2	2	-	-	-	-
200	120	6	1	-	-	-	-	1
210	130, 200	2	1	-	2	-	-	-
220	210	4	1	-	-	-	-	-
230	220	1	2	2	-	-	1	-
240	30	4	3	-	-	-	-	-
250	240	9	4	-	-	-	-	-
260	240	9	4	-	-	-	-	1
270	210, 260	6	4	-	-	-	-	-
280	40	1	-	2	-	-	-	-
290	260	4	2	-	-	-	-	-

Without the constraint of resources a time analysis using the CPM technique is required to solve the unlimited resource leveling problem. Figure 3 shows the precedence diagram for a project example drawn using the detail information of the activities tabulated in Table 1.

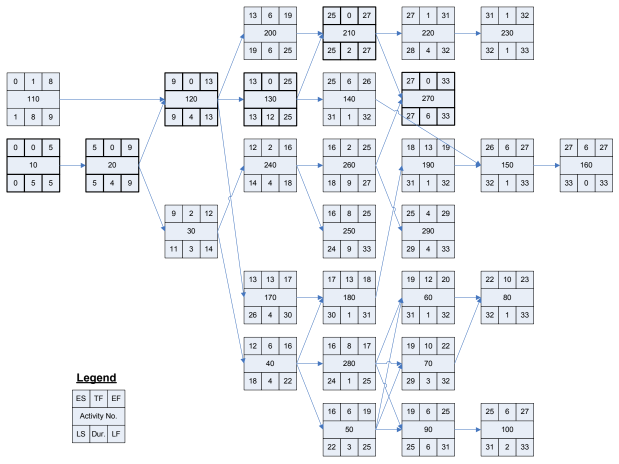

Figure 3. Precedence diagram for a project instance

Figure 3 also shows all schedule data and a legend is placed inside Figure 3. In detail, the bold line indicates the critical path of the project. Durations, the early start time (EST), the early finish time (EFT), the late start time (LST), the late finish time (LFT), and the total float (TF) values calculated by forward and backward calculations were shown. The critical path time is 33 weeks, which is the project duration fixed for solving the resource leveling problem. For the sake of simplicity, the network is assumed to have only one type of resource and each activity can consume only one type of resource, i.e., labor. Let us assume that a limited amount of labor available for the completion of a project instance is set to 10 laborers for the case example only. A resource histogram based on the CPM early start is drawn in Figure 4.

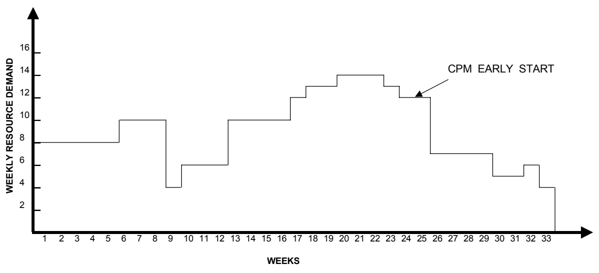

Figure 4. Resource histogram produced by CPM early start

4.4 Comparison with heuristic methods

A project example was used to verify the mechanism of the GA-based resource leveling model. The resource histograms for leveling using both the least total float rule and the GA-based model are depicted in Figure 5. The amount of resource available for each activity up to day 16 does not exceed the required amount of resources in this specific example project. On day 17, the resource conflict occurs between activities 250 and 260. Activity 250 that has the negative future float value of 8 is first shifted because its future float is less than that of activity 260, which has the negative value of 2. In the next time frame, day 26, a resource conflict occurs again among activities 100, 140, and 290. Three activities have the positive future float value of 4, 4, and, 2, respectively and activity 290, which has the least future float value, is then shifted.

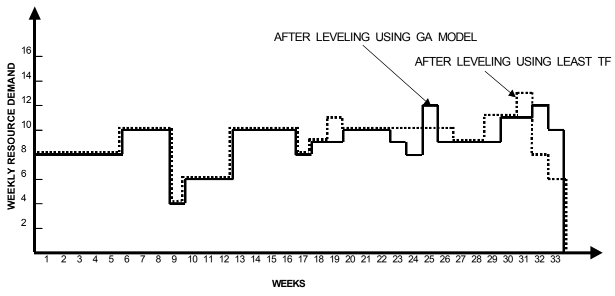

Figure 5. Comparison of resource profiles produced by heuristic rule and GA

As shown in Figure 5, two resource histograms have shown similar numbers of violations resulting in resource overutilization because of the fixed duration, meaning that the project duration cannot exceed the fixed duration of 33 weeks. It is observed that the histogram obtained using the algorithm proposed was shown to be the same as, or very close to that produced by the existing resource leveling based on the least total float rule, which shifts non-critical activities individually. The notable thing about the GA-based resource leveling model is that it can provide several equally good scheduling alternatives, even though they are not included in Figure 5.

5. CONCLUSIONS AND RECOMMENDATIONS

This paper presents a GA-based optimal algorithm for a resource leveling model that simultaneously levels the resources of a set of non-critical activities experiencing conflicts simultaneously up to an assumed level of resource rates specified by the planner using a pair-wise comparison of the activities being considered. A parameter called the future float value is adopted and applied as an indicator for assigning leveling priorities to the sets of activities experiencing conflicts. A construction project network example was worked out to show the performance of the proposed model. The histogram obtained using the algorithm proposed was shown to be the same as, or very close to that produced by the existing resource leveling method based on the least total float rule, which shifts non-critical activities individually.

Future work is recommended to consider the effect of the resource leveling method on the duration of the project. It is possible to expand the project duration in a situation where a contractor for the project does not care to lengthen the project duration but needs to balance resources up to an optimal level of resource rates, even though this situation is not considered in this paper. In this case, any activity being postponed might reduce the remaining total float to the project network.

REFERENCES

[1] Chan, W., Chua, D. K. H., and Kannan, G. (1996). "Construction Resource Scheduling with Genetic Algorithms." *J. Constr. Engrg. and Mgmt.*, ASCE, Vol. 122, No. 2, pp. 125-132.

[2] Shanmuganayagam, V. (1989). "Current Float Techniques for Resource Scheduling." *J. Constr. Engrg. and Mgmt.*, ASCE, Vol. 115, No. 3, pp. 401-411.

[3] Hegazy, T. (1999). "Optimization of Resource Allocation and Leveling Using Genetic Algorithms." *J. Constr. Engrg. and Mgmt.*, ASCE, Vol. 125, No. 3, pp. 167-175.

[4] Leu, S., and Yang, C. (1999). "GA-Based Multicriteria Optimal Model for Construction Scheduling." *J. Constr. Engrg. and Mgmt.*, ASCE, Vol.125, No. 6, pp. 420-427.

[5] Senouci, A. B., and Eldin, N. N. (2004). "Use of Genetic Algorithms in Resource Scheduling of Construction Projects," *J. Constr. Engrg. and Mgmt.*, ASCE, Vol. 130, No. 6, pp. 869-877.

[6] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass.

[7] Whitley, D. (1993). *A Genetic Algorithms Tutorial.* Retrieved on April 18, 2004 from http://samizdat.mines.edu/ga_tutorial/.

[8] Harris, R. B. (1978). *Precedence and Arrow Networking Techniques for Construction*, John Wiley & Sons, Inc., New York, N.Y.

[9] Sathyanarayana, K., Rajeev, S., Kalyanaraman, V. (1993). "Optimum resource allocation in construction projects using genetic algorithms," *Proc. of 3rd Int. Conf. on the Application of AI to Civ. and Struct. Engrg.*, Edinburgh, England.

[10] Moselhi, A., and Lorterapong, P. (1993). "Least Impact Algorithm for Resource Allocation." *Can. J.Civ. Eng.*, CSCE, Vol. 20, No. 2, pp. 180-188.

[11] Bean, J. C. (1994). "Genetic Algorithms and Random Keys for Sequencing and Optimization," *J. on Computing*, Operations Research Society of America, Vol. 6, No. 2, pp. 154-160.

[12] Li, H., Cao, J., and Love, P. E. D. (1999). "Using Machine Learning and GA to Solve Time-Cost Trade-Off Problems." *J. Constr. Engrg. and Mgmt.*, ASCE, Vol. 125, No. 5, pp. 347-353.

[13] Willis, R. J., and Hastings, N. A. J. (1976). "Project Scheduling with Resource Constraints Using Branch and Bound Methods," *Operations Research*, 27 (2), pp. 341-349.