APPLYING ELITIST GENETIC ALGORITHM TO RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM

Jin-Lee Kim, Ph.D.[1] and Ok-Kyue Kim, Ph.D.[2]
[1] Assistant professor, Department of Engineering Technology, Missouri Western State University, 4525 Downs Drive, St. Joseph, MO 64507, U.S.A.
[2] Professor, Department of Architectural Engineering, Chungbuk National University, 12 Gaeshin-dong, Heungduk-gu, Cheongju, Chungbuk, 361-763, Korea

Abstract

The objective of this research study is to develop the permutation-based genetic algorithm for solving the resource-constrained project scheduling problem in construction engineering by incorporating elitism into genetic algorithm. A key aspect of the algorithm was the development of the elitist roulette selection operator to preserve the best individual solution for the next generation so the improved solution can be obtained. Another notable characteristic is the application of the parallel schedule generation scheme to generate a feasible solution to the problem. Case studies with a standard test problem were presented to demonstrate the performance and accuracy of the algorithm. The computational results indicate that the proposed algorithm produces reasonably good solutions for the resource-constrained project scheduling problem.

Keywords: Algorithm, Heuristics, Optimization, Project management, Resource-constrained project scheduling

1. Introduction

The demand of project scheduling software has continued to grow at an annual rate of almost 20% [1]. Project scheduling software packages often consider constrained resources. However, their capability to solve resource-constrained problems is either fragile or nonexistent [2]. Thus, there is a remarkable need for efficient solution approaches that allow for the complexities of real-world problems, which is an intended contribution of this research.

The resource-constrained project scheduling problem (hereinafter RCPSP) have been solved with the various exact methods, priority-rule based heuristics, and various meta-heuristic methods. First, the various exact methods employ some form of mathematical programming such as dynamic programming and zero-one programming or other analytical procedure such as implicit enumeration with branch and bound to search for the best possible solutions. 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. Considerable progress has been made to produce optimal results by depending on strong assumptions for small-sized project networks. However, no optimal procedures have proven to be computationally feasible for large, complex projects that can occur in practice [3]. Therefore, heuristic and meta-heuristic approaches are needed for large-sized project networks.

Second, priority-rule based heuristics employ some rule of thumb or experience to determine priorities among activities competing for available resources. They combine one or more priority rules and schedule generation scheme to generate one or more schedule. These heuristic procedures generally produce solutions for the RCPSP in a reasonable amount of time, even though the size of the project network is large. However, they have proven to be inconsistent with regard to the quality of results produced on project networks [3]. Finally, various meta-heuristic methods, such as genetic algorithm (GA), simulated annealing (SA), tabu search (TS), and ant colonies (AC), have been also applied to the RCPSP to overcome the drawbacks of the exact optimal methods and priority-rule based heuristics and to improve the performance of the existing meta-heuristic methods.

The genetic algorithm (GA), a meta-heuristic and optimization technique, has emerged as a tool that is beneficial for a variety of study fields including construction applications since the introduction in the 1960's by Holland [4]. GA has also 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 [5]. GA has been used successfully to solve construction management problems, including resources scheduling with a small number of activities [6, 7, 8, 9].

A permutation-based genetic algorithm proposed by Hartmann makes use of activity list representation [10]. The study also proposed additional two encodings, which include priority value based genetic algorithm similar to the work of Lee and Kim [11] and priority-rule based genetic algorithm similar to the work of [12]. From their computational results, the permutation-based encoding genetic algorithm outperformed two other encoding algorithms.

This paper presents a new permutation-based elitist genetic algorithm (hereinafter PEGA) for solving the RCPSP in construction engineering. The proposed algorithm aims to allocate multiple available construction resources to activities of a single project to achieve the objective of minimizing the project duration. A key aspect of the algorithm was the development of the elitist roulette selection operator to preserve the best individual solution for the next generation so the improved solution can be obtained. Parallel schedule generation scheme was applied to generate a feasible solution to the problem. Case studies with a standard test problem were presented to demonstrate the performance and accuracy of the algorithm over other GA method under single and multiple resources.

2. Objective function of RCPSP

The RCPSP aimed to allocate the available resources to activities so as to find the shortest duration of a project within the constraints of precedence relationships. The assumptions underlying this problem were that the availability of resources is constrained to some maximum value, and that the project has to be completed using the given resources. As a result of the RCPSP, a schedule that shows the shortest duration with resource limits was created for a project network. The objective function was formulated for a permutation-based elitist genetic algorithm for the RCPSP. As a constrained optimization problem, the RCPSP belongs to one type of sequencing problem. Therefore, the objective function for the algorithm is to minimize the project duration when constrained by precedence relationships among project activities and the availability of resources.

3. Applying elitist into permutation-based genetic algorithm

The main procedure of the PEGA was developed and implemented in this paper, as shown in Figure 1. Several operators were used in the development of the algorithm. They include (1) the random number generator for producing an initial population, (2) the parallel schedule generation scheme for calculating a fitness value of each individual, (3) the elitist roulette wheel selection operator for selecting a parent individual for the next generation, (4) the one-point crossover operator for exchanging parent string segments and recombining them to produce two resulting offspring individuals, and finally (5) the uniform mutation operator for playing a role of random local search which searches regardless of the direction of learning to obtain the better solution. PEGA algorithm is terminated if it meets either the number of generation or time to stop specified in the start phase.

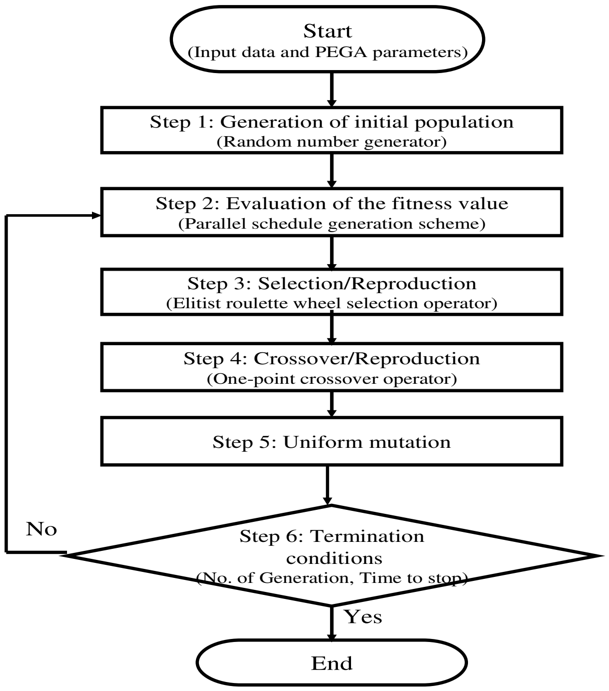
Start
(Input data and PEGA parameters)
Step 1: Generation of initial population
(Random number generator)
Step 2: Evaluation of the fitness value
(Parallel schedule generation scheme)
Step 3: Selection/Reproduction
(Elitist roulette wheel selection operator)
Step 4: Crossover/Reproduction
(One-point crossover operator)
Step 5: Uniform mutation
No
Step 6: Termination conditions
(No. of Generation, Time to stop)
Yes
End

Figure 1: Main procedure of PEGA

PEGA was developed by employing four basic operators: elite selection [13], roulette wheel selection [4], one-point crossover [10], and uniform mutation [5]. The initial population of possible solutions to the RCPSP was created to apply the algorithm in the very first step of global search. A fitness value of an individual in the initial population was calculated, using the parallel schedule generation scheme proposed by Kelley [14] and the one of Brooks [15]. The selection of the parent individuals was made through the elitist roulette wheel selection operator for the next generation. The elitist roulette wheel selection operator is the combined operator using the elite selection and the roulette wheel selection. Using the parent individuals obtained from the selection operator, one-point crossover operator was performed by exchanging parent individual segments and then recombining them to produce two resulting offspring individuals. The uniform mutation operator was performed to play the role of random local search.

3.1 Permutation-based encoding

A schedule has to be represented to encode the RCPSP. In addition to the schedule representation, a schedule generation scheme needs to decode the schedule representation into a schedule. A schedule representation is a representation of a priority-structure among the activities. A solution for the RCPSP was represented in a chromosome that represented an activity sequence for the problem. A chromosome is also called an individual that was given by an activity sequence. Each gene in a chromosome stands for an activity number. An activity has a lower priority than all preceding activities in the sequence and a higher priority than all succeeding activities. Thus, an individual becomes precedence feasible permutation of the set of activities because an activity cannot comes after the position of one of its successors (predecessors) in the list used for the generation of an individual. A precedence feasible permutation was generated using random number generator developed in this research.

This research adopted a permutation-based encoding that was appropriate for solving the RCPSP [10, 16]. An initial population composed of precedence feasible individuals was produced by the random number generator. It is notable that the random number generator simply provides precedence feasible solutions, but does not give the fitness value (the project duration), a possible starting and finishing time of an activity, and the feasibility of resource constraints. Random number generator, for example, generates an individual {2, 7, 1, 6, 4, 3, 8, 9, 5, 11, 10} for 11 non-dummy activities.

Worth noting is that the fitness function is different from the objective function for the clarification of a computation process of the fitness value. As mentioned previously, the objective function is to minimize the fitness function, which generates the fitness value of a project throughout the scheme process. The fitness function is to find the maximum value out of all fitness values of every activity to be scheduled in a project. The maximum value is obtained by comparing the finish time of the last activity and the fitness value of the activity just before the last activity. The parallel schedule generation scheme proposed by Kelley [14] and the one of Brooks [15] was utilized to calculate the fitness value of an individual. When all activities in an individual are scheduled, the fitness value is obtained from the maximum value between the finish time of the last activity and the fitness value of the activity just before the last activity.

The purpose of applying the parallel schedule generation scheme to the individuals in a population was to obtain schedules that showed the resource profile and the project duration. It is important to note that a uniquely determined schedule (phenotype) computed using the parallel scheme can be related more than one individual (genotype). A uniquely determined schedule means that it is possible for several individuals to have the same fitness value, but their starting time should be totally different. The unique schedules in the search space as genotypes may be related to the same schedule, which is the project duration for the RCPSP.

The objective function for the RCPSP is to minimize the fitness function. Therefore, a transformation from the minimization function to a maximization function was needed to ensure that the individual selection process performs correctly. This research modified the transformation method proposed by Lee and Kim [11] by adding a new parameter named transformation power (TP) to their transformed objective function in order to emphasize low fitness values [17]. The addition of the transformation power showed how much the transformed fitness values can be emphasized by making the original fitness values low.

3.2 Combining Elitist with Roulette Wheel Selection

The elitist preserving selection called elitism proposed by De Jong [13] was adopted to combine with the roulette wheel selection operator in this research. The elitist roulette selection is operated using the procedure shown in Figure 2. Elitism first preserves the best individual generated up to generation t into the current generation t+1, if the fitness value of an individual in the current population is larger than that of every individual in the current population. The roulette wheel selection operator developed by Holland [4] has been employed, as used in many studies [7, 10]. The concept of the selection was to determine selection probability for each individual proportional to the fitness value.

```
ELITIST ROULETTE SELECTION OPERATOR PROCEDURE

1: EliteChrome = best Chrome in current generation
2: Create a transformed fitness for each chrome
3: Create summation of these fitnesses
4: For each chrome J in the new generation
5:  seed = RandomSeed[0,sum of transformed fitness]
6:  for Chrome I = second chrome to the last chrome_
      in current generation
7:      if sum of transformed fitness from Chrome 1 to_
          I > seed
8:        Chrome J = Chrome before I
9:  current generation = new generation
10: current generation's first Chrome = EliteChrome
```

Figure 2: Procedure of elitist roulette selection operator

3.3 One-Point Crossover Operator

The goal of a crossover operator is to combine pieces of information coming from different individuals in the population. It is important to preserve good building blocks and maintain randomness and population diversity for searching non-redundant solutions [16]. The order of the first several activities is the key to preserve the whole individual, providing the basis for the remaining activities and deciding how good the order of the remaining activities will be to a certain degree.

The one-point crossover operator is capable of preserving schemata in a more effective manner because it keeps the first half of both parents intact and is less random than UX3. The probability of disrupting short defining length is rather low, even though crossover operation in the beginning of an individual is likely to disrupt schema [18]. The rationale for using the one-point crossover operator is that a precedence feasible offspring is generated if it is applied to precedence feasible parents [10, 16]. The theorem was proven by Hartmann [10].

3.4 Uniform Mutation Operator

The goal of the uniform mutation is to exchange two neighboring genes without violating precedence relationship in order to create an individual that could not have been produced by the crossover operator. The uniform mutation operator was operated as follows: for each individual from a generation, the operator generates a real random number and then swaps an activity after pivot point with activity at pivot point if a random number is equal to or less than mutation probability. The operator can be ineffective because the genes in neighboring individual positions could be switched while still representing the same schedule. Therefore, it is important to note that a mutation on an individual does not necessarily change the related schedule because interchanging two activities that have the same start time in the activity sequence is likely to change the individual, but not the related schedule.

4. Results and analysis

PEGA was programmed using the JAVA programming language on the Windows XP operation system, and Microsoft® Office Excel 2003 was selected as the representation and analysis tool for the data. The parameters of the algorithm include population size, transformation power, crossover probability, and mutation probability for global search. Two different types of termination conditions can be determined to stop the run of the algorithm. They include the number of generations and timeout. Two output options were available for the analysis of data. They include full output and summary output.

In order to test and verify the overall procedure of the PEGA, a case example of a small construction project schedule was extracted from the work of Shanmuganayagam [19]. Figure 3 shows the example of schedule network, which includes activity name, duration, and resource requirements.

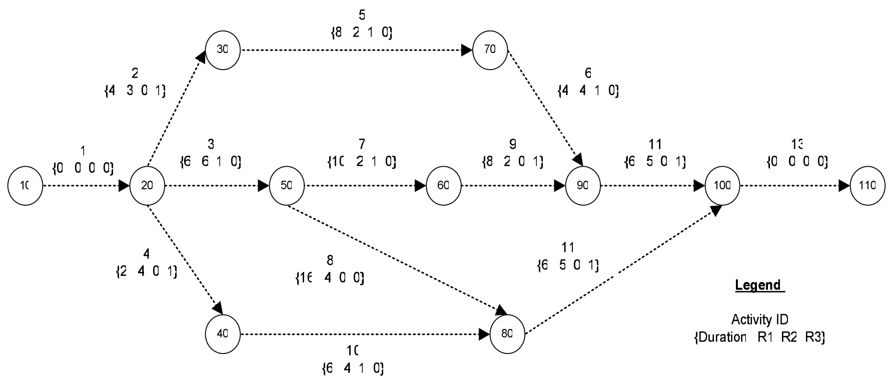

Figure 3: Example of schedule network

After preliminary studies, the parameter configurations were determined to obtain the best performance from the algorithm because the population size and various parameters were the critical elements for optimal performance. This research tested the algorithm on the example using both single and multiple resources in order to demonstrate the robustness of the current approach.

4.1 Effect of Elitist on the Performance

This section describes the effect of elitist on the performance of the algorithm by running PEGA on the case example of 11 non-dummy activities with single resource. The default set of the parameters as follows: the population size, transformation power, crossover and mutation probability were set to 30, 1.6, 0.5, and 0.03, respectively. The algorithm was terminated with the number of generation of 100 using the parallel scheme. Figure 4 shows the profile of the schedule obtained from the elitist individual, which was produced from the last generation of 100. The project duration was found at 38 days, which can be considered near-optimal solution to the example problem.

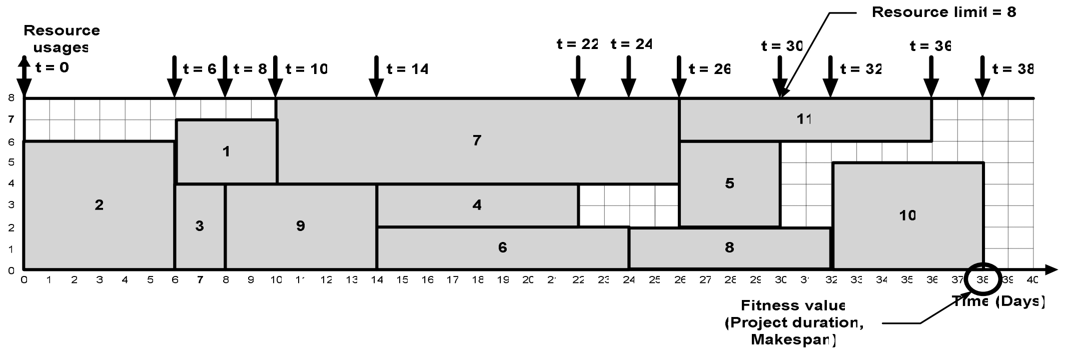

Figure 4: Profile of the schedule produced by the elitist

4.2 Comparison with other GA

The case example was used to verify the mechanism of the PEGA. All activities of the project network were scheduled using just one resource with a fixed resource profile to make an impartial comparison with the results obtained from the work of Chan et al. [6]. The population size is set to 50 and the total number of generation was set to 40 experimental runs so the total trial size of 2,000 was performed. After many trials, crossover rate and mutation rate were set to 0.5 and 0.03, respectively.

Table 1 shows the various schedules in comparison to three schedules produced by GA-scheduler [6]. The algorithm produced the project duration of 38 days, which is same as those obtained by the GA-scheduler. It also generated 1,426 unique schedules, which amounts to 71.30% of the total schedules of 2000. It took the total CPU time of 610 milliseconds for the algorithm to solve the RCPSP with single resource. Worth noting is that the algorithm is able to provide several equally good and feasible scheduling alternatives, which indicate the similar result to GA-scheduler [6]. The remarkable thing about the algorithm is that it does not require applying any type of penalty factor since the schedules were uniquely determined by the algorithm. The algorithm does not also depend on any set of heuristic rules.

Table 1: Comparison of Various Schedules by Method

Activity		Starting times of activities obtained by						
No.	Resource requirement	GA-scheduler [6]			PEGA (This research)			
		S1	S2	S3	Elitist	S1	S2	S3
1	3	6	7	8	6	6	6	6
2	6	0	0	0	0	0	0	0
3	4	10	11	14	6	10	20	6
4	2	15	15	19	14	12	10	10
5	4	28	28	28	26	28	28	26
6	2	6	7	8	14	12	10	8
7	4	6	6	6	10	6	6	10
8	2	17	18	18	24	22	22	24
9	4	32	32	32	8	22	22	18
10	5	22	22	22	32	32	32	32
11	2	28	28	28	26	28	28	26

4.3 Scheduling Project with Multiple Resources

PEGA was run to take into account of the multiple resources to the same case example. Three different types of resources were considered. It is obvious that when multiple resources are required, project duration will make changes, depending on the resource availability and requirements. As the case of single resource, the resource availabilities are constant over the project duration and the resource availabilities of three resources are assumed to be 8, 1, and 1, respectively.

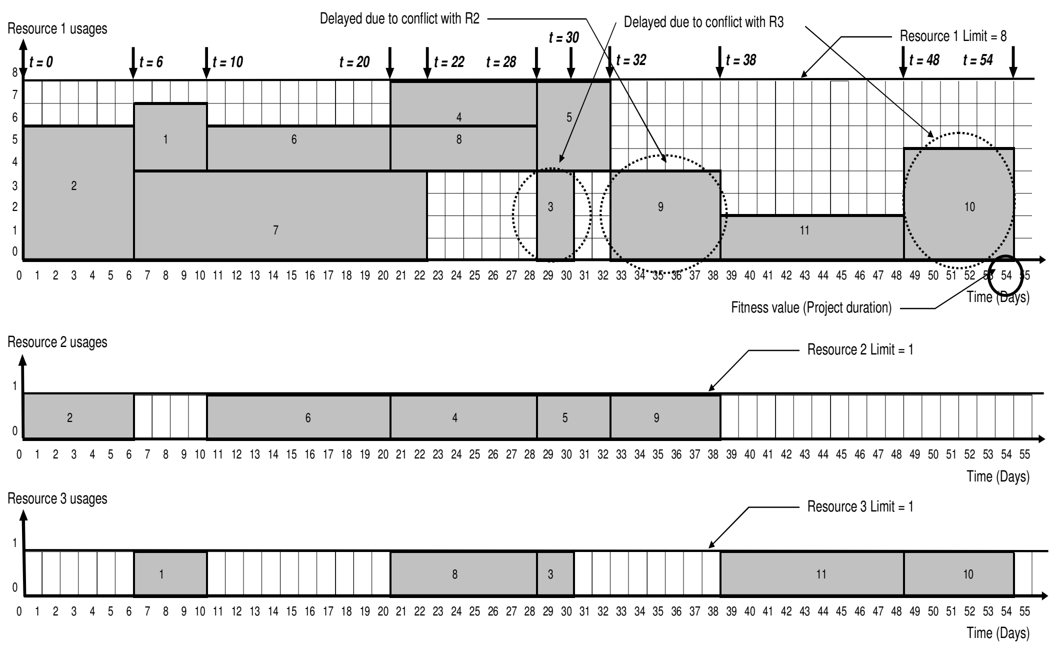

Figure 5: Scheduling with multiple resources

After many trials, the population size, crossover rate, and mutation rate were set to 50, 0.5, and 0.03, respectively. The overall fitness value, which is the project duration of the individual considered, was obtained for multiple resources using the parallel schedule generation scheme. Figure 5 shows the result of scheduling the case example with multiple resources. The project duration was 54 days that was obtained by the partial schedule. As a result of scheduling with multiple resources, it was found that activity 9 was delayed for two days due to conflict with R2. It was also found that activities 3 and 10 were postponed for 6 and 10 days due to resource conflicts with R3, respectively.

5. Conclusion

This paper addressed the permutation-based elitist genetic algorithm for solving multiple resource-constrained project scheduling problem. Compared with other genetic algorithm method, the algorithm showed that the elitist preserves the best individual solution for the next generation so the improved solution can be obtained. It was also found that the algorithm is able to explore and exploit several near-optimal solutions, which may include the optimal solution, because heuristic methods do not generally provide the optimal solution. The algorithm is useful to solve the multiple resource-constrained project scheduling problem. Several equally good scheduling alternatives generated by the algorithm provide more information for decision making than the only one schedule produced by the heuristic method.

References

[1] **Wallace, R., and Halverson, W. (1992).** "Project management: A critical success factor or a management fad." *Industrial Engineering*, 24(4), 48-50.

[2] **De Wit, J., and Herroelen, W. (1990).** "An evaluation of microcomputer-based software packages for project management." *European Journal of Operations Research*, 49, 102-139.

[3] **Kim, J.-L., and Ellis, R. D. (2005).** "A Framework for Integration Model of Resource-Constrained Scheduling using Genetic Algorithms," *Proceedings of the 2005 Winter Simulation Conference*, M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds., December 4-7, Orlando, FL, 2119-2126 (CD-Rom).

[4] **Holland, J. K. (1975).** Adaptation in Neural and Artificial Systems, University of Michigan Press, Ann Arbor, MI.

[5] **Sakawa, M. (2002).** Genetic Algorithms and Fuzzy Multiobjective Optimization. Kluwer Academic Publishers, Norwell, Massachusetts.

[6] **Chan, W., Chua, D. K. H., and Kannan, G. (1996).** "Construction Resource Scheduling with Genetic Algorithms." *Journal of Construction Engineering and Management, ASCE*, 122(2): 125-132.

[7] **Leu, S., and Yang, C. (1999).** "GA-Based Multicriteria Optimal Model for Construction Scheduling." *Journal of Construction Engineering and Management, ASCE*, 125(6): 420-427.

[8] **Hegazy, T. (1999).** "Optimization of Resource Allocation and Leveling Using Genetic Algorithms." *Journal of Construction Engineering and Management, ASCE*, 125(3): 167-175.

[9] **Hegazy, T., and Kassab, M. (2003).** "Resource Optimization Using Combined Simulation and Genetic Algorithms." *Journal of Construction Engineering and Management, ASCE*, 129(6): 698-705.

[10] **Hartmann, S. (1998).** "A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling." *Naval Research Logistics*, 45: 733-750.

[11] **Lee, J.-K., and Kim, Y.-D. (1996).** "Search Heuristics for Resource-Constrained Project Scheduling." *The Journal of the Operational Research Society*, 47(5): 678-689.

[12] **Dorndorf, U., and Pesch, E. (1995).** "Evolution Based Learning in a Job Shop Scheduling Environment." *Computers and Operations Research*, 22: 25-40.

[13] **De Jong, K. A. (1975).** "An Analysis of the Behavior of a Class of Genetic Adaptive Systems." Ph.D. Dissertation, University of Michigan, Ann Arbor, Mich.

[14] **Kelley, J. E. Jr. (1963).** "The Critical-Path Method: Resources Planning and Scheduling." In J. F. Muth and G. L. Thompson (Eds.), Industrial Scheduling, Prentice-Hall, New Jersey, 347-365.

[15] **Bedworth, D. D., and Bailey, J. E. (1982).** Integrated Production Control Systems-Management, Analysis, Design, Wiley, New York.

[16] **Zhuang, M., and Yassine, A. A. (2004).** "Task Scheduling of Parallel Development Projects using Genetic Algorithm." *Proceedings of ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Salt Lake City, Utah USA, September 28-October 2, 2004, 1-11.

[17] **Kim, J.-L. (2006).** "A multiheuristic approach to resource constrained project scheduling: an adaptive hybrid genetic algorithm" Ph.D. Dissertation, University of Florida, Gainesville, FL.

[18] **Goldberg, D. E. (1989).** Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

[19] **Shanmuganayagam, V. (1989).** "Current Float Techniques for Resource Scheduling." *Journal of Construction Engineering and Management, ASCE*, 115(3): 401-411.