Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

PERMUTATION-BASED ELITIST GENETIC ALGORITHM USING SERIAL SCHEME FOR LARGE-SIZED RESOURCE-CONSTRAINED PROJECT SCHEDULING

Jin-Lee Kim

Department of Engineering Technology
4525 Downs Drive, Missouri Western State University
St. Joseph, MO 64507, U.S.A.

ABSTRACT

This research paper presents a new permutation-based Elitist genetic algorithm using serial schedule generation scheme for solving a large-sized multiple resource-constrained project scheduling problem, which is one of the most challenging problems in construction engineering. A key aspect of the algorithm was the application of the elitist roulette selection operator to preserve the best individual solution for the next generation so as to obtain the improved solution. Serial schedule generation scheme was applied to generate a feasible solution to the problem. Results for large-sized project network problems were presented to demonstrate the performance and accuracy of the algorithm. The computational results indicate that the proposed algorithm not only produces reasonably good solutions for the resource scheduling problem over the heuristic method and other GA, but also able to solve large-sized multiple resource-constrained project scheduling problems applicable to the construction industry.

1 INTRODUCTION

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. No optimal procedures have proven to be computationally feasible for large, complex projects that can occur in practice (Moselhi and Lorterapong 1993). 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 (serial or parallel) 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 (Hegazy 1999).

Recently, various meta-heuristic methods, such as genetic algorithm (GA), simulated annealing (SA), tabu search (TS), and ant colonies (AC), have been 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. Among these method, the 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 (Holland 1975). Several studies have been done to solve the RCPSP using GA (Hartmann 1998, Kohlmorgen et al. 1999, Alcaraz and Maroto 2001, Hindi et al. 2002, Toklu 2002). GA has also been used successfully to solve construction management problems, including resources scheduling with a small number of activities (Chan et al. 1996, Leu and Yang 1999, Hegazy and Kassab 2003). A permutation-based GA proposed by Hartmann (1998) makes use of activity list representation. The study also proposed additional two encodings, which include priority value based GA similar to the work of Lee and Kim (1996) and priority-rule based GA similar to the work of Dorndorf and Pesch (1995). From their computational results, the permutation-based encoding GA outperformed two other encoding algorithms.

The demand of project scheduling software has continued to grow at an annual rate of almost 20% (Wallace and Halverson 1992). Project scheduling software packages often consider constrained resources, but their capability to solve RCPSP is either fragile or nonexistent (De Wit and Herroelen 1990). Thus, there is a need for efficient solution approaches that allow for the complexities of real-world problems, which is an intended contribution of this research. This paper presents a new permutation-based Elitist genetic algorithm using serial schedule generation scheme for solving large-sized multiple RCPSP in construction engineering. 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 as to obtain the improved solution. Serial schedule generation scheme (hereinafter SGS) was applied to generate a feasible solution to the problem. Several large-sized project network problems were solved to demonstrate the performance and accuracy of the algorithm over either a heuristic method or other GA method under the constraints of single and multiple resources and to verify the capability and efficiency of the Elitist genetic algorithm.

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.

2 PERMUTATION-BASED ELITIST GENETIC ALGORITHM

The main procedure of the permutation-based Elitist genetic algorithm using serial SGS was shown in Figure 1.

<u>MAIN PROCEDURE</u>

```
Start procedure
INPUT
  BeginTime = currentTime
   Perform Random Number Generation Procedure
   CurrentGeneration = 0
   While CurrentGeneration < NumGenerations AND_
         currentTime-BeginTime < AllowedTime
     Calculate Fitness for All Individuals
     Perform Elitist Roulette Selection Procedure
     Current Generation = CurrentGeneration + 1
     Perform One-point Crossover Procedure
     Perform Uniform Mutation Procedure
  EndTime = currentTime
  TotalTime = EndTime - BeginTime
OUTPUT
End procedure
```

Figure 1: Pseudocode for Elitist genetic algorithm

Several operators employed in the development of the Elitist genetic algorithm include (1) the random number generator for producing an initial population, (2) the serial SGS 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.

2.1 Encoding and Decoding for the RCPSP

A schedule has to be represented to encode the RCPSP. In addition to the schedule representation, a SGS 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 come 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 (Hartmann 1998, Zhuang and Yassine 2004). An initial population composed of precedence feasible individuals was produced by the random number generator (Kim 2006). 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 serial SGS proposed by Kelley (1963) was utilized to calculate the fitness value of an individual. It consists of the n stages, which is the same as the number of activities to be scheduled. Associated with each stage, a set of activity to be scheduled can be classified into two disjoint activity sets: scheduled set and decision set. The activities that were already scheduled are in the scheduled set, and they belong to the partial schedule. The decision set contains the unscheduled activities with every predecessor being in the scheduled set. One activity is selected according to the order of the activity list representation at each stage, and then scheduled at its earliest precedence and resource feasible start time. Afterwards, the selected activity is removed from the decision set and put into the scheduled set. 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 serial SGS to the individuals in a population was to obtain schedules that showed the resource profile and the project duration. A uniquely determined schedule (phenotype) computed using the serial SGS can be related to 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.

2.2 Combining Elitist with Roulette Wheel Selection

The elitist preserving selection called elitism proposed by De Jong (1975) was adopted to combine with the roulette wheel selection operator. The elitist roulette selection is operated using the procedure shown in Figure 2.

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: Pseudocode of elitist roulette selection operator

Elitism first preserves the best individual generated up to generation t in 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 (1975) was used, as in many GA studies. The concept of the selection was to determine selection probability for each individual proportional to the fitness value.

2.3 One-Point Crossover and Uniform Mutation Operators

Two different types of crossover operators, union crossover operator 3 (UX3) (Leu and Yang 1999) and one-point crossover (Hartmann 1998), were identified as good methods for the permutation-based encoding for the solution to the RCPSP. They were developed to deal with this type of ordering problem that occurred due to crossover operation. The one-point crossover is operated using the procedure shown in Figure 3.

ONE-POINT CROSSOVER PROCEDURE

```
1: Select two parent individuals from a population individuals
2:  Select the same position along both parents using_
      a random integer
3:  Select the first half sub-individuals from parent 1 and _
      parent 2 (They are named sub-individual 1 and 2)
4:  Find the exclusive sub-individuals from parent 2 for_
      Sub-individual 1 and from parent 1 for sub-individual 2_
      (They are named sub-individual 3 and 4, respectively)
5:  Place sub-individual 1 into unfixed positions at the_
      beginning of offspring 1.
6:  Place sub-individual 2 into unfixed positions at the_
      beginning of offspring 2.
7:  Position sub-individual 3 into unfixed positions_
      following Sub-individual 1 in offspring 1
8:Position sub-individual 4 into unfixed positions following_
      Sub-individual 2 in offspring 2
```

Figure 3: Pseudocode of one-point crossover operator

UX3 fixes character duplication problem after the chromosome operation as well as maintains precedence relationships by creating two exclusive sub-individuals from parent individuals and then randomly writing the elements directly to the offspring individuals and then taking accounts of activity precedence relationships when writing characters from sub-individuals into the offspring individuals. However, the performance of the UX3 operator was reduced since it caused significant change to the individual representations of the parent individuals by disrupting potential building blocks and high fitness schema at each generation (Zhuang and Yassine 2004). Their findings are reasonable because if better solutions were found with

UX3, it is not due to the recombination theories fundamental to the GA, but to the randomization of UX3. The major disadvantage of UX3 operator is that it must change gene positions more frequently than the one-point crossover operator. The same issue can be raised when applying the one-point crossover operator, but one-point crossover preserves the good schemas by keeping the first half of activities intact (Reeves 1995, Hartmann 1998). For this reason, the one-point crossover operator was selected for the permutation-based encoding to the RCPSP.

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. 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.

3 EXPERIMENTAL RESULTS AND ANALYSIS

Elitist genetic algorithm 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, crossover probability, and mutation probability for global search. The algorithm terminates when the pre-specified number of generations is met.

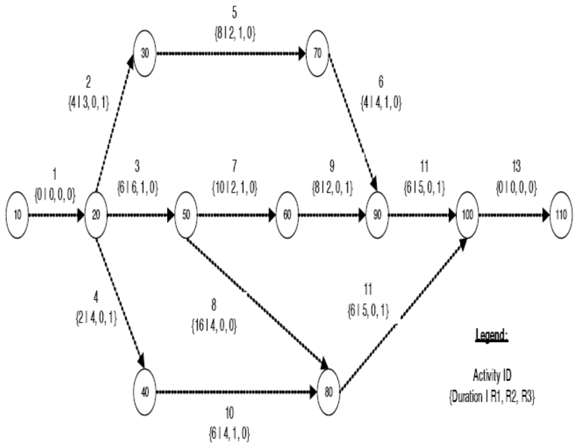

Figure 4: Example of schedule network

In order to test the overall procedure of the algorithm, a case example of a small construction project schedule was extracted from the work of Shanmuganayagam (1989). Figure 4 shows the schedule network, which includes activity name, duration, and resource requirements.

3.1 Effects of Elitist Roulette Selection Operator on the Performance

This section describes the effects of the elitist roulette selection operator on the performance of the Elitist genetic algorithm. The default set of the parameters as follows: the population size, crossover and mutation probability were set to 30, 0.5, and 0.03, respectively. The algorithm was terminated with the number of generation of 100 using the serial SGS. Figure 5 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.

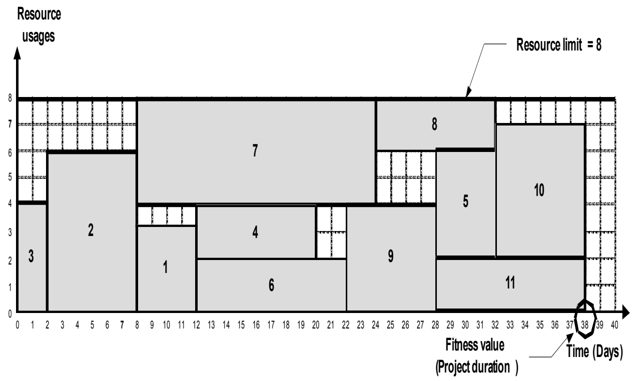

Figure 5: Profile of the schedule produced by the elitist

3.2 Comparison with Heuristic Methods and Other GAs

The case example was also used to verify the mechanism of the Elitist genetic algorithm. 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. (1996). 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. The crossover and mutation rates were set to 0.5 and 0.03, respectively. Table 1 shows the various schedules in comparison to the single schedule obtained by the heuristic rule (Shanmuganayagam 1989) and three schedules produced by GA-scheduler (Chan et al. 1996). Elitist genetic algorithm produced the project duration of 38 days, which is same as those obtained either by the heuristic rule or by the GA-scheduler.

Table 1: Comparison of Various Schedules by Method

Activity		Starting times of activities obtained by							
No.	Resource requirement	Heuristic method (Shanmuganayagam 1989)	GA-scheduler (Chan et al. 1996)			*Elitist genetic algorithm* (this research)			
		S1	S1	S2	S3	Elitist	S1	S2	S3
1	3	6	6	7	8	8	6	8	6
2	6	0	0	0	0	2	0	0	0
3	4	10	10	11	14	0	10	6	6
4	2	10	15	15	19	12	12	12	10
5	4	28	28	28	28	28	28	28	26
6	2	6	6	7	8	12	12	12	10
7	4	6	6	6	6	8	6	6	8
8	2	16	17	18	18	24	22	22	20
9	4	32	32	32	32	22	22	22	20
10	5	22	22	22	22	32	32	32	32
11	2	28	28	28	28	28	28	28	26

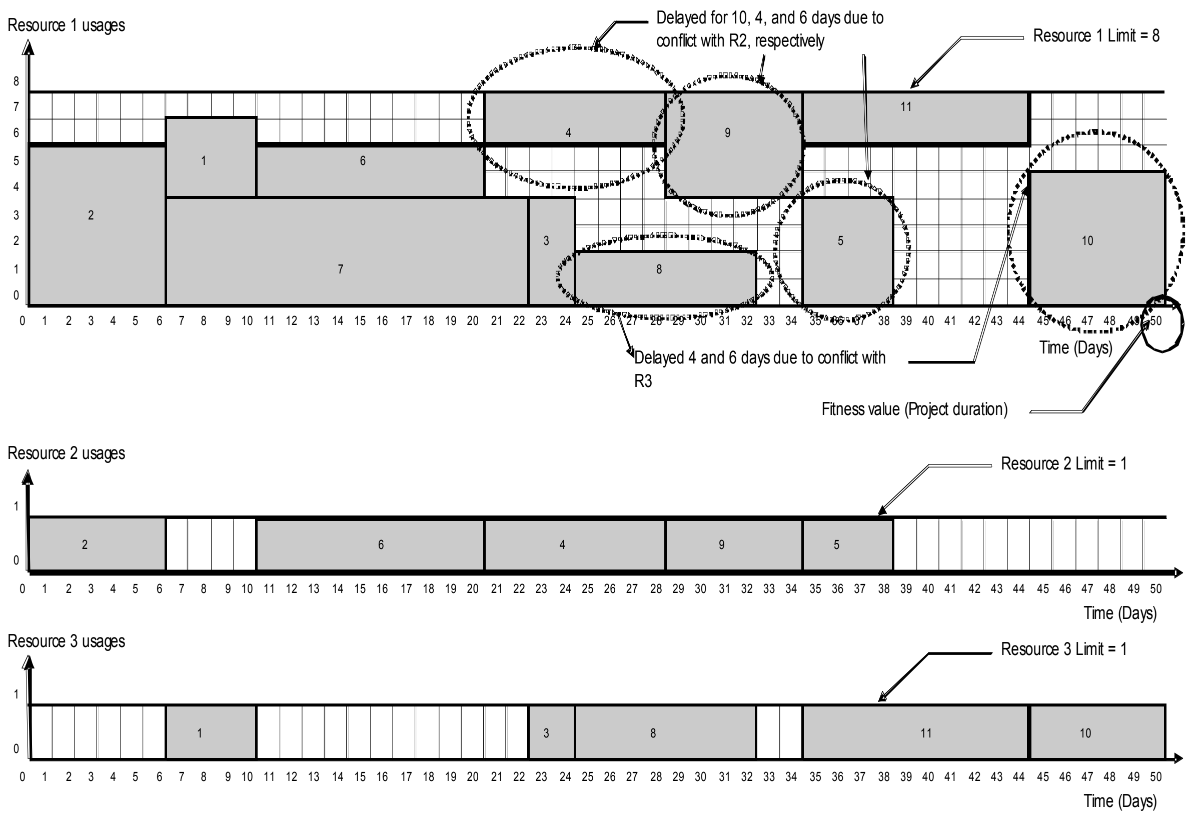

Figure 6: Scheduling procedure for multiple resources

It also generated 1,387 unique schedules, which amounts to 69.35% of the total schedules of 2000. It took the total CPU time of 531 milliseconds for the algorithm to solve the RCPSP with single resource. Elitist genetic algorithm is able to provide several equally good and feasible scheduling alternatives, which indicate the similar result to GA-scheduler (Chan et al. 1996). The result is reasonable in that there is a significant difference between the genetic algorithm approach and the heuristic approach (Leu et al.

1999). Elitist genetic algorithm 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.

3.3 Scheduling Project with Multiple Resources

Elitist genetic algorithm was run to take into account of the multiple resources to the same case example. Three different types of resources were considered. 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. 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 serial SGS. Figure 6 shows the result of scheduling the case example with multiple resources. The project duration was 50 days because the finish time (50 days) of the last activity (activity 10) is greater than the fitness value (44 days) of the activity (activity 11) just before the last activity. As a result of scheduling with multiple resources, it was found that activities 4, 9, and 5 were delayed for 10, 4, and 6 days due to resource conflicts with R2, respectively. It was also found that activity 8 and 10 were postponed for 4 and 6 days due to resource conflicts with R3, respectively.

3.4 Scheduling Large-sized Projects with Multiple Resources

An experiment was conducted to verify the performance of the Elitist genetic algorithm. Three large-sized project scheduling problems obtained from the PSPLIB (Kolisch and Sprecher 1996) were used in this experiment. The projects consist of 30, 60, and 120 activities, respectively. For each problem size, a problem instance has four renewable resources. The overall performance of the Elitist genetic algorithm was measured by the means of finding the best fitness value, which can be considered a near-optimal solution to the RCPSP. The input parameter values for the algorithm were set as follows: Initial population size, crossover rate, mutation rate were set to 100, 0.5, and 0.03, respectively. The termination condition was set to the maximum number of generations of 100. Table 2 shows the minimum fitness values, total algorithm runtime in millisecond (ms), and the number of unique schedules as a result of scheduling three large-sized multiple RCPSP. Elitist genetic algorithm found 43, 77, and 119 minimum fitness values for 30-Activity, 60-Activity, and 120-Activity, respectively, as they converges to a single point across the number of generation. The optimal solution for the problem with 30 non-dummy activities is known (Demeulemeester and Herroelen 1997), while for the problem instances with 60 and 120 non-dummy activities, only heuristic solutions, which are lower bound solutions (Klein and Scholl 1999 for 60-Activity and Brucker and Knust 2003 for 120-Activity), are known. The algorithm required more time to solve a larger problem than a smaller one as expected. It also generated 3,490, 3,483, and 3,478 unique schedules, which amounts to 69.8%, 69.7%, and 69.6% of the total schedules of 5000, respectively.

Table 2: Results for Scheduling Large-sized Projects

Project size	30-Activity	60-Activity	120-Activity
Minimum fitness (Makespan)	43	77	119
Optimality	43	77	99
Total algorithm run-time (ms)	23,797	23,641	23,297
No. of unique schedules	3,490	3,483	3,478

4 CONCLUSIONS

This paper introduced a permutation-based Elitist genetic algorithm using serial schedule generation scheme for solving a large-sized multiple RCPSP. Compared with a heuristic method and other GA method, the developed Elitist genetic algorithm produces reasonably good solutions for the RCPSP. The computational results based on several large-sized multiple RCPSP indicate that the proposed algorithm not only provides several equally good scheduling alternatives but also is capable to solve large-sized RCPSP within a reasonable time. Several equally good scheduling alternatives generated by the proposed algorithm will provide more information for decision making than the only one schedule produced by the heuristic method.

For the improvement of the Elitist genetic algorithm application, two point or other crossover methods may also be employed on a random basis to improve the integrity of building blocks, even though Elitist genetic algorithm used only one-point crossover to apply the recombination theories fundamental to the GA. The use of the elitism in GA is to ensure that small populations do not lose the current best solution in the population. However, roulette wheel selection has an exponential selection pressure which drives a solution to convergence prior to sufficient exploration of the solution space. The tournament selection can be applied to overcome the stochastic nature of the selection process in the Elitist genetic algorithm since the selection pressure of pair-wise tournament selection is consistent regardless of the contents of the population.

REFERENCES

Alcaraz, J., and Maroto, C. 2001. A Robust Genetic Algorithm for Resource Allocation in Project Scheduling. Annals of Operations Research, 102: 83-109.

Brucker, P., and Knust, S. 2003. Lower Bounds for Resource-Constrained Project Scheduling Problems. European Journal of Operations Research, 149, 302-313.

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.

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.

Demeulemeester, E. L., and Herroelen, W. S. 1997. New Benchmark Results for the Resource-Constrained Project Scheduling Problem. Management Science, 43(11): 1485-1492.

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.

Dorndorf, U., and Pesch, E. 1995. Evolution Based Learning in a Job Shop Scheduling Environment. Computers and Operations Research, 22: 25-40.

Hartmann, S. 1998. A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling. Naval Research Logistics, 45: 733-750.

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

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.

Hindi, K. S., Yang, H., and Fleszar, K. 2002. An Evolutionary Algorithm for Resource-Constrained Project Scheduling. IEEE Transactions on Evolutionary Computation, 6(5): 512-518.

Holland, J. K. 1975. Adaptation in Neural and Artificial Systems, University of Michigan Press, Ann Arbor, MI.

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.

Kim, J.-L. 2006. A Multiheuristic Approach to Resource Constrained Project Scheduling: An Adaptive Hybrid Genetic Algorithm. Ph.D. Dissertation, Department of Civil and Coastal Engineering, University of Florida, Gainesville, FL

Klein, R., and Scholl, A. 1999. Computing Lower Bounds by Destructive Improvement: An Application to Resource-Constrained Project Scheduling. European Journal of Operational Research, 112: 322-346.

Kohlmorgen, U., Schmeck, H., and Haase, K. 1999. Experiences with Fine-Grained Parallel Genetic Algorithms. Annals of Operations Research, 90: 203-219.

Kolisch, R., and Sprecher, A. 1996. PSPLIB - A Project Scheduling Problem Library. European Journal of Operational Research, 96: 205-216.

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.

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.

Leu, S., Chen, A., and Yang, C. 1999. Fuzzy Optimal Model for Resource-Constrained Construction Scheduling. Journal of Computing in Civil Engineering, 13(3): 207-216.

Moselhi, O., and Lorterapong, P. 1993. Least Impact Algorithm for Resource Allocation. Canadian Journal of Civil Engineering, CSCE, 20(2): 180-188.

Reeves, C. R. 1995. Genetic Algorithms and Combinatorial Optimization. Applications of Modern Heuristic Methods, V. J. Rayward-Smith (ed.), Alfred Waller, Henley-on-Thames, 111-125.

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

Toklu, Y. C. 2002. Application of Genetic Algorithms to Construction Scheduling With or Without Resource Constraints. Canadian Journal of Civil Engineering, 29: 421-429.

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

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.

AUTHOR BIOGRAPHIES

JIN-LEE KIM is an assistant professor in the Department of Engineering Technology at Missouri Western State University. His research interests include information technology in construction, simulation based resource scheduling and optimization techniques. He is a member of ASCE. His web page can be found via <www.missouriwestern.edu>.