Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

EXAMINING THE RELATIONSHIP BETWEEN ALGORITHM STOPPING CRITERIA AND PERFORMANCE USING ELITIST GENETIC ALGORITHM

Jin-Lee Kim

California State University, Long Beach
1250 Bellflower Boulevard
Long Beach, CA 90840, USA

ABSTRACT

A major disadvantage of using a genetic algorithm for solving a complex problem is that it requires a relatively large amount of computational time to search for the solution space before the solution is finally attained. Thus, it is necessary to identify the tradeoff between the algorithm stopping criteria and the algorithm performance. As an effort of determining the tradeoff, this paper examines the relationship between the algorithm performance and algorithm stopping criteria. Two algorithm stopping criteria, such as the different numbers of unique schedules and the number of generations, are used, while existing studies employ the number of generations as a sole stopping condition. Elitist genetic algorithm is used to solve 30 projects having 30-Activity with four renewable resources for statistical analysis. The relationships are presented by comparing means for algorithm performance measures, which include the fitness values, total algorithm runtime in millisecond, and the flatline starting generation number.

1 INTRODUCTION

A genetic algorithm is a potential meta-heuristic that is sufficiently powerful to implement an evolutionary strategy for solving difficult problems such as construction resource scheduling, for example, the resource-constrained project scheduling problem. A remarkable advantage of the general genetic algorithm is that it travels in a search space with more possible solutions (Goldberg 1989). This advantage is of main concern for many researches although it can be difficult to choose an encoding and fitness function. On the other hand, it is necessary to improve computational time to overcome a great length of time for the required computation.

A genetic algorithm generally starts 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 such as selection, crossover, and mutation. The procedure continues until the termination condition is satisfied. The fitness values follow the uniform distribution with mean equal to the solution to the problem and appropriately zero standard deviation, starting from a random distribution with mean and unknown standard deviation at the beginning of the generation.

The success of using the general genetic algorithm depends on how quickly and accurately it converges to the optimal solution while avoiding local minima, as it requires a length of time to reach to the optimal solution. Many studies reveal that a major disadvantage of the general genetic algorithm is that it requires a relatively large amount of computational time to search for the solution space before the solution is finally attained (Espinoza, Minsker, and Goldberg 2005; Kim 2009).

The Elitist strategy can be considered to be an effective way of improving the performance of the genetic algorithm applications. Taranenko and Vesel (2001) developed a new genetic algorithm for the maximum independent set problem based on the elitist strategy and concluded that the effectiveness of the algorithm was very satisfactory. Costa and Oliveira (2004) presented an elitist genetic algorithm for multiobjective optimization in engineering problems and proved that elitism led to a good compromise between computational time and size of the elite population. Ahn et al. (2004) proposed a memory-efficient elitist genetic algorithm for solving hard optimization problems quickly and effectively. Geroa, Garcíaa, and del Coz Díazb (2005) modified the elitist genetic algorithm to apply it into the design optimization of complex steel structures and showed that the elitist genetic algorithm is able to work with complex structures under different load and constraint conditions. Majumdar and Bhunia (2007) presented an elitist genetic algorithm to solve a generalized assignment problem with imprecise costs and times and reported that the algorithm selected better chromosomes for the next generation in each generation. Kim and Ellis (2008) presented a permutation-based elitist genetic algorithm to solve construction resource scheduling problems.

Thus, it is necessary to identify the tradeoff between the algorithm stopping criteria and the algorithm performance. As an effort of determining the tradeoff, this paper examines the relationship between the algorithm performance measures and algorithm stopping criteria. Elitist genetic algorithm (Elitist GA) developed in the previous phases of this research is used to solve 30 project networks having 30-Activity with four renewable resources for statistical analysis. This paper uses two algorithm stopping criteria, such as the different numbers of unique schedules and the number of generations, while many existing GA studies employ the number of generations as a sole stopping condition. The main objective of this paper is to present the relationships for the trade-off in terms of the fitness values, total algorithm runtime in millisecond, and the flatline starting generation number. The statistical analysis using the completely randomized design is conducted in this paper.

The following section briefly introduces Elitist GA for resource scheduling problem and its stopping criteria, followed by the descriptions of data and variables for running Elitist GA in order to obtain data. A computational experiment using the completely randomized design is presented, immediately followed by statistical results and analysis. This paper then makes concluding remarks that summarize the findings and recommend the future study with regard to the usage of the number of uniquely determined schedules as a stopping condition.

2 ELITIST GENETIC ALGORITHM AND ITS STOPPING CRITERIA

This section presents Elitist GA and its stopping criteria to determine the relationship between the algorithm performance measures, which include the fitness value, the flatline starting generation number, total algorithm runtime in millisecond, and algorithm stopping criteria. First, it briefly introduces the algorithm to solve construction resource scheduling problem, followed by the brief overview of genetic operators such as elitism with roulette wheel, one-point crossover, and uniform mutation operators. It then describes the algorithm stopping criteria such as the number of generations, which is the most commonly used condition, and the number of unique schedules, which is the advanced condition for comparison purpose.

2.1 ELITIST GA FOR CONSTRUCTION RESOURCE SCHEDULING

Elitist GA for solving the resource-constrained project scheduling problem was developed to search for an optimal solution to the problem using four GA operators: selection by elitism, roulette wheel selection, one-point crossover, and uniform mutation. The operators and their functions include the random number generator for producing an initial population and examining the precedence feasible individuals, the serial schedule generation scheme for computing a fitness value of each individual, the elitist roulette wheel selection operator for selecting a parent individual for the next generation when copying the best chromosome to the new population, the one-point crossover operator for exchanging parent string segments and recombining them to produce two resulting offspring individuals, and finally the uniform mutation opera-

tor for playing a role of random local search which searches regardless of the direction of learning to obtain the better solution (Kim and Ellis 2008). Elitist GA is used here to compare algorithm stopping criteria because it is designed to select the options for stopping criteria and because Elitist operator improves the performance of the general GA without the elitist operator.

2.2 ALGORITHM STOPPING CRITERIA FOR ELITIST GA

Many studies selected the generation number as their sole stopping condition. As an effort of exploring a way to reduce a lengthy computation time of genetic algorithms, the author selected the number of uniquely determined schedule to identify the trade-off between algorithm stopping criteria and algorithm performance measures. Elitist GA for construction resource scheduling were originally designed to select one of three different algorithm stopping criteria, which include the number of generations, the number of uniquely determined schedules, and algorithm runtime. The uniquely determined schedule means that it is possible for several individuals to have the same fitness value that is equal to the project duration, but their starting time should be totally different. Figure 1 shows the example of two individuals, No. 8 and No. 9, whose overall fitness values are equal to 42 days, but their starting time of each activity are different. The unique schedules as genotypes in the search space may be related to the same schedule.

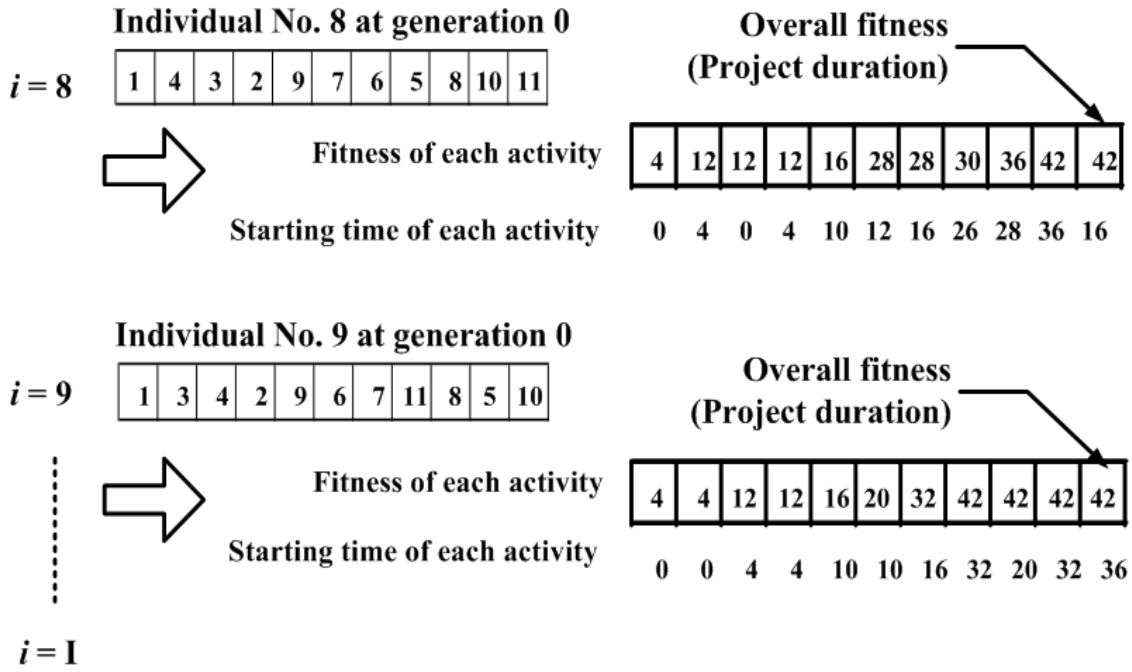

Figure 1: A uniquely determined schedule (After Kim 2009)

3 DATA AND VARIABLES

Thirty instance projects are selected from the PSPLIB for the computational experiment. The PSPLIB, the project scheduling problem library generated by Kolisch and Sprecher (1996), is the source that is now widely used in the resource scheduling study area. Each of 30 projects consists of 30 activities having four renewable resources. Each problem has been generated using a full factorial design of parameters, which determine the characteristics of the resource and precedence constraints. It is notable that the optimal solutions for the projects with 30 activities are known so that the fitness values obtained from Elitist GA using the five different algorithm stopping criteria can be compared for optimality. Table 1 tabulates the parameter values for Elitist GA. Considering the computational costs and memory requirements, the

parameter values are set to 100, 1.6, 0.5, and 0.03 for population size, transformation power, crossover probability, and mutation probability, respectively.

Algorithm stopping criteria considered here is divided into two criteria, the generation number of 100 and the number of unique schedules. Further, the number of unique schedule is divided into four criteria such as 1000, 3000, 5000, and 1000 schedules because it may affect the computational time to reach the optimal solution. The rationale of the four different unique schedule criteria is to see how many unique schedules are equivalent to the number of generations of 100. Five algorithm stopping criteria are used in this experiment. For example, if 100_Gen is used as a stopping condition, the algorithm terminates with the number of generations of 100, which means the algorithm generates 10,000 trials. If 1000_Sch is used as a stopping condition, the algorithm terminates when it produces 1000 uniquely determined schedules as mentioned previously. In order to examine the behaviors of the algorithm according to the stopping criteria, the algorithm performances are measured using four different unique schedules, such as 1000, 3000, 5000, and 10000, and tested against the optimality, in addition to the number of generations of 100.

Table 1: Elitist GA Parameter Values

Parameter Name	Symbol	Value/Meaning
Population size	*Pop_size*	100
Transformation power	*TP*	1.6
Crossover probability	*Cp*	0.5
Mutation probability	*Mp*	0.03
Five stopping criteria	*100_Gen*	100 number of generation
	1000_Sch	1000 unique schedules
	3000_Sch	3000 unique schedules
	5000_Sch	5000 unique schedules
	10000_Sch	10000 unique schedules

4 EXPERIMENTS AND RESULTS

A computational experiment was designed to determine the relationship between the algorithm performance measures and algorithm stopping criteria. The performance of Elitist GA algorithm is measured by comparing the project durations produced from the algorithm against those obtained from optimality available from PSPLIB. The algorithm runtime and the flatline starting generation number are chosen to compare the computational times depending on the stopping criteria. A flatline starting generation point is the first point out of the generation number specified as an input of Elitist GA, where the fitness values to the problem converges to a single number with the standard deviation of zero.

The experiments consist of three comparison tests, which include the fitness value, the algorithm runtime, and the flatline starting generation number, varying the different algorithm stopping criteria. The completely randomized design is used here as it is the simplest experimental design for comparing more than two population means. The variable of interest is the dependent or response variable, and the independent variables are called factors. A treatment is a factor level in an experiment with one factor. The analysis of variance is used to determine whether a factor affects the response variable. If the factor is significant, the mean response differs for the various treatments (Meyer and Krueger 1998; Kuehl 2000).

4.1 Comparison of Fitness Values

The author employs pairwise comparisons using the Tukey procedure to compare each of performance measure means with each of the other measure means. The parameters of interest here are all pairwise differences among the performance measure means. The pairwise comparison aims to detect significant inequalities for all performance measure means. The first parameter of interest is the difference among the means of fitness values obtained by running Elitist GA according to five algorithm stopping criteria. Ta-

ble 2 tabulates the fitness values for all 30 project instances by algorithm stopping criteria, in addition to the comparison against their optimality values that are available from PSPLIB.

Table 2: Comparison of Fitness Values by Stopping Criteria

Project ID	Optimality	Stopping Criteria				
		100_Gen	1000_Sch	3000_Sch	5000_Sch	10000_Sch
1	43	43	45	46	43	45
2	47	51	48	47	48	47
3	47	47	47	47	47	47
4	62	62	62	62	62	62
5	39	39	39	39	39	39
6	48	49	48	49	48	48
7	60	60	60	60	60	60
8	53	54	53	55	53	53
9	49	50	50	50	49	50
10	45	45	45	45	45	45
11	38	38	38	38	38	38
12	51	51	51	51	51	51
13	43	43	43	43	43	43
14	43	43	43	43	43	43
15	51	51	51	51	51	51
16	47	47	47	47	47	47
17	47	47	47	47	47	47
18	54	54	54	54	54	54
19	54	54	54	54	54	54
20	43	43	44	43	43	43
21	72	72	72	72	72	72
22	40	40	40	40	40	40
23	57	57	57	57	57	57
24	98	98	98	98	98	98
25	53	53	53	53	53	53
26	54	54	54	54	54	54
27	48	48	48	48	48	48
28	54	54	54	54	54	54
29	65	65	65	65	65	65
30	59	59	59	59	59	59

One-way analysis of variance is provided using Minitab® 15 to test whether the mean ratios of fitness values according to the five algorithm stopping criteria differ. The null and alternative hypotheses are H_o: $\mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5$ and H_a: at least two means differ, where μ_i is the mean fitness value for i^{th} stopping criterion. The Tukey 95% simultaneous confidence intervals for all pairwise comparisons among the five stopping criteria show that the null hypothesis is not rejected because the observed significance level or p-value of 1.0 is greater than $\alpha = 0.05$. Thus, we do not have sufficient evidence to conclude that the true mean ratios differ for at least two of the five stopping criteria. The confidence intervals also indicate that the means may not differ. In the case of the fitness values, all five intervals overlap, so it is not possible that the corresponding mean ratios differ significantly. The findings indicate that the stopping criteria of both the number of generations and the four different numbers of unique schedules are significantly meaningful at searching for the solution to the resource scheduling problem.

4.2 Comparison of Computational Time

The second parameter of interest is the difference among the means of algorithm runtimes obtained by running Elitist GA depending on five algorithm stopping criteria. Figure 2 shows the algorithm runtimes in millisecond for the individual 30 projects and the average of all 30 projects, respectively. The findings clearly show that there is a significant difference among the five algorithm stopping criteria in terms of algorithm runtime, as one can easily expect.

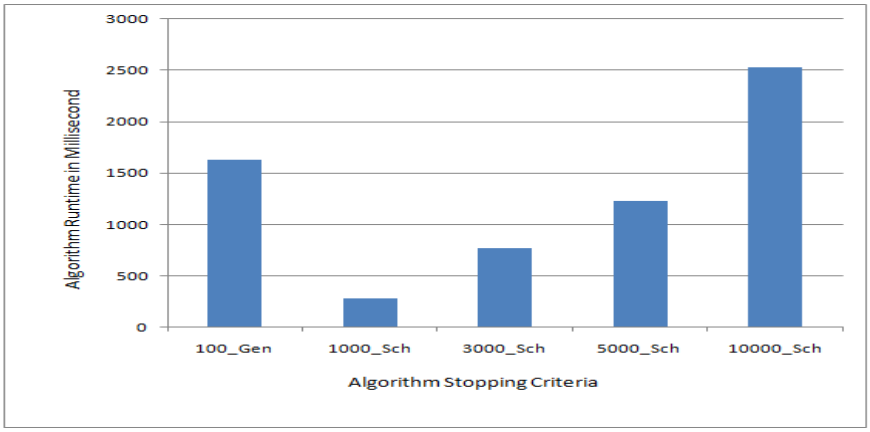

Figure 2: Average algorithm runtimes in millisecond for 30 projects

For the comparison test of the difference among the means of algorithm runtimes by stopping criteria, one-way analysis of variance is provided to test whether the mean ratios of algorithm runtimes according to the five algorithm stopping criteria differ. The null and alternative hypotheses are H_o: $\mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5$ and H_a: at least two means differ, where μ_i is the mean algorithm runtime for i^{th} stopping criterion. The test statistics, F=2240.62, has a p-value of 0. The p-value is the probability of obtaining a test statistic as large as F value, assuming Ho is true. Since p-value is less than $\alpha = 0.05$, the null hypothesis is rejected. Thus, we have sufficient evidence to conclude that the true mean ratios differ for at least two of the five algorithm stopping criteria. Indeed, the Tukey 95% simultaneous confidence intervals for all pairwise comparisons among the five algorithm stopping criteria show that the null hypothesis is rejected because the observed significance level or p-value of 0.0 is less than $\alpha = 0.05$. Thus, we have sufficient evidence to conclude that the true mean ratios differ for at least two of the five algorithm stopping criteria. The confidence intervals also indicate that the means may differ. In the case of the algorithm runtime, more than two intervals overlap, which means that it is possible that the corresponding mean ratios differ significantly. The findings indicate that the stopping criteria of the number of generations and the four different numbers of unique schedules do significantly affect the algorithm runtime to run Elitist GA for the resource scheduling problem. Therefore, the further intensive statistical analysis is required for the algorithm runtime to determine the tradeoff by identifying the equivalent number of unique schedules associated with the number of generation.

4.3 Comparison of Flatline Starting Generation

The third and last parameter of interest in this paper is the difference among the means of the flatline starting generation numbers obtained by running Elitist GA according to the five algorithm stopping criteria. Figure 3 shows the comparison of the average flatline starting generation numbers for all 30 projects. This figure clearly shows that there is a significant difference among the five algorithm stopping criteria in terms of the flatline starting generation number, as one can easily expect.

For the comparison test of the difference among the means of the flatline starting generation numbers by stopping criteria, one-way analysis of variance is provided to test whether the mean ratios of the flat-

line starting generation number according to the five algorithm stopping criteria differ. The null and alternative hypotheses are H_o: $\mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5$ and H_a: at least two means differ, where μ_i is the mean flatline starting generation number for i^{th} stopping criterion. The test statistics, F=16.14, has a p-value of 0.0. Since p-value is less than $\alpha = 0.05$, the null hypothesis is rejected. Thus, we have sufficient evidence to conclude that the true mean ratios differ for at least two of the five algorithm stopping criteria. Indeed, the Tukey 95% simultaneous confidence intervals for all pairwise comparisons among the five algorithm stopping criteria show that the null hypothesis is rejected because the observed significance level or p-value of 0.0 is less than $\alpha = 0.05$. Thus, we have sufficient evidence to conclude that the true mean ratios differ for at least two of the five algorithm stopping criteria. The confidence intervals also indicate that the means may differ. In the case of the flatline starting generation number, more than two intervals overlap, so it is possible that the corresponding mean ratios differ significantly. The findings indicate that the stopping criteria of both the number of generations and the four different numbers of unique schedules do significantly affect the flatline starting generation number for solving the resource scheduling problem. Therefore, further intensive statistical analysis is required for the flatline starting generation number to determine the tradeoff between the equivalent number of the flatline starting generation number and the associated number of generations.

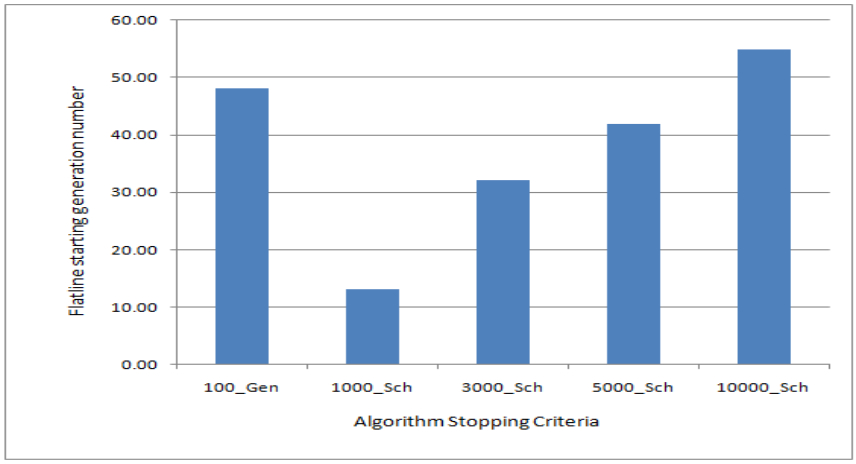

Figure 3: Average flatline starting generation numbers by stopping criteria

Based on the experimental results presented here, we conclude that no significant differences exist among the means of fitness values in terms of the algorithm stopping criteria. Whether it is either the number of generations or the number of uniquely determined schedules, it is evident that the fitness values obtained by running Elitist GA well match the optimality as shown in Table 2. The algorithm runtime and the flatline generation number are the factors that are affected by the different algorithm stopping criteria. For the cases of the algorithm runtime and the flatline starting generation number, both the algorithm runtime and the flatline starting generation number increase as the number of the uniquely determined schedule increases, as shown in Figures 2 and 3.

5 CONCLUDING REMARKS

This paper presents the results using the completely randomized design to examine the relationship between the genetic algorithm stopping criteria and the algorithm performance measures. Elitist GA for solving construction resource scheduling is used. Five algorithm stopping criteria were used to terminate the algorithm. The algorithm performance measures, which include the fitness value, the total algorithm runtime in millisecond, and the flatline starting generation number, were compared to test whether their means differ. The results support that the uniquely determined schedule is important in that it does affect

the algorithm runtime and the flatline starting generation number. Also, it is meaningful to further examine how many number of unique schedules are equivalent to a certain number of generation. For future study, intensive statistical analysis is necessary to determine the tradeoff between algorithm runtime and the number of unique schedules. The ultimate goal of this study is to propose the baseline stopping criteria for an arbitrary algorithm to solve construction resource scheduling problems by using the unique schedule so that algorithm designers can use a certain number of unique schedules to examine if their algorithms produce optimal and/or near-optimal solutions under the same conditions. Because the null hypotheses for both algorithm runtime and the flatline starting generation number considered here are rejected, it is possible to further test pairwise comparisons using the least significant difference method, which controls the weak sense experiment-wise error rate.

REFERENCES

Ahn, C. W., K. P. Kim, and R.S. Ramakrishna. 2004. A memory-efficient elitist genetic algorithm. *Lecture Notes in Computer Science*, Springer Berlin, Heidelberg, 3019:552-559.

Costa, L., and P. Oliveira. 2004. An elitist genetic algorithm for multiobjective optimization. *Metaheuristics: computer decision-making (Applied Optimization)*, ed. M. G. Resende and J. Pinho de Sousa, 217-236. Kluwer Academic Publishers, Norwell, MA, USA,.

Espinoza, F. P., B.S. Minsker, and D.E. Goldberg. 2005. Adaptive hybrid genetic algorithm for groundwater remediation design. *Journal of Water Resources Planning and Management* 131(1): 14-25.

Geroa, M. B. P., A.B. Garcíaa, and J.J. del Coz Díazb. 2005. A modified elitist genetic algorithm applied to the design optimization of complex steel structures. *Journal of Constructional Steel Research*, 61(2), 265-280.

Goldberg, D. E. 1989. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

Kim, J.-L. 2009. Improved genetic algorithm for resource-constrained scheduling of large projects. *Canadian Journal of Civil Engineering* 36(6):1016-1027.

Kim, J.-L., and R. D. Ellis. 2008. Permutation based elitist genetic algorithm for multiple resource constrained project scheduling problem. *Journal of Construction Engineering and Management* 134(11): 904-913.

Kolisch, R., and A. Sprecher. 1996. PSPLIB - A project scheduling problem library. *European Journal of Operational Research*, 96:205-216.

Kuehl, R. O. 2000. *Design of experiment: Statistical principles of research design and analysis*. 2nd ed., Duxbury Press, Pacific Grove, CA.

Majumdar, J., and A.K. Bhunia. 2007. Elitist genetic algorithm for assignment problem with imprecise goal. *European Journal of Operational Research*, 177(2):684-692.

Meyer, R., and D. Krueger. 1998. *A Minitab guide to statistics*. Prentice Hall, Upper Saddle River, NJ.

Taranenko, A., and A. Vesel. 2001. An elitist genetic algorithm for the maximum independent set problem. *Proceedings of 23rd International Conference on Informational Technology Interfaces, ITI 2001*, Pula, Croatia, 373-378.

AUTHOR BIOGRAPHIES

JIN-LEE KIM, Ph.D., P.E., is an assistant professor of Dept. of Civil Engineering & Construction Engineering Management at California State University, Long Beach. He received his BE and first ME degrees in Architectural Engineering from Chungbuk National University, Korea and his second ME and PhD in Civil Engineering from University of Florida. He is a registered professional engineer in Florida. His research interests include information technology in construction, simulation-based resource scheduling, optimization techniques, and sustainable design and construction. He is a member of ASCE. His email address is <jin5176@gmail.com>.