Efficient Elitist Genetic Algorithm for Resource-Constrained Project Scheduling

Kim, Jin-Lee*

Abstract

This research study presents the development and application of an Elitist Genetic Algorithm (Elitist GA) for solving the resource-constrained project scheduling problem, which is one of the most challenging problems in construction engineering. Main features of the developed algorithm are that the elitist roulette selection operator is developed to preserve the best individual solution for the next generation so as to obtain the improved solution, and that parallel schedule generation scheme is used to generate a feasible solution to the problem. The experimental results on standard problem sets indicate that the proposed algorithm not only produces reasonably good solutions to the problems over the heuristic method and other GA, but also can find the optimal and/or near optimal solutions for the large-sized problems with multiple resources within a reasonable amount of time that will be applicable to the construction industry. This paper will help researchers and/or practitioners in the construction project scheduling software area with alternative means to find the optimal schedules by utilizing the advantages of the Elitist GA.

Keywords: Resources, scheduling, optimization, genetic algorithms, heuristics, project management

1. Introduction

1.1 Research Background

The critical path method (CPM) has been used widely in the construction industry. However, it has proven to be helpful only when the project deadline is not fixed and the resources are not constrained by either availability or time. Since there are limitations on the quantities of available resources in practice, construction companies are confronted with the problem of using resources efficiently within the project constraints. Therefore, resource scheduling arises as a result of problems with the availability of resources. In this paper, resource allocation, which is well known as the resource-constrained project scheduling problem (hereinafter RCPSP), is used to describe the scheduling technique. The RCPSP has 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). Thus, heuristic and meta-heuristic approaches are needed for large-sized project networks.

*일반회원, Assistant Professor, Department of Engineering Technology, Missouri Western State University, U.S.A. Email: jkim@missouriwestern.edu.

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 to 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).

Finally, 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 methods, 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 (1975), followed by Goldberg (1989).

1.2 Existing Studies on RCPSP with GA

GA has been used successfully to solve construction management problems, including resource scheduling with a small number of activities (Chan et al. 1996; Hegazy 1999; Leu et al. 1999; Leu and Yang 1999; Toklu 2002; Hegazy and Kassab 2003). Chan et al. (1996) developed GA-scheduler applicable to the RCPSP in the construction industry. They made use of the serial scheme for allocating resources into an activity. Hegazy (1999) proposed improved heuristics with GAs for applications to resource allocation and leveling problems. Leu et al. (1999) developed a fuzzy model for the resource-constrained construction schedule. Leu and Yang (1999) presented a GA-based multicriteria optimal model for construction scheduling. Toklu (2002) developed a GA that applies directly on schedules using a vector of start times. Hegazy and Kassab (2003) examined resource optimization using combined simulation and GAs.

Several studies have been also done to solve the standard RCPSP using a GA (Lee and Kim 1996; Hartmann 1998 and 2002; Kohlmorgen et al. 1999; Alcaraz and Maroto 2001; Hindi et al. 2002; Valls et al. 2005). Most meta-heuristics that employed the activity list representation in the RCPSP literature make use of the serial scheme as a decoding procedure. However, a few meta-heuristics employed the parallel scheme as a decoding procedure for the RCPSP. Several works of research for the RCPSP conducted by Brucker et al. (1998), Demeulemeester and Herroelen (1997), and others have been considered to be the currently most powerful exact procedures for solving the standard RCPSP. However, their procedures do not have a capability to find optimal solutions, which are optimal schedules, for project networks with 60 activities or more. According to the work of Hartmann (1998), an experimental evaluation study shows that commercial software packages for project management provide schedules with an average deviation of 4.3-9.8% from the optimal solution for a project network with up to 30 activities.

1.3 Research Objectives

This research study was motivated to find the optimal solutions to the RCPSP by considering the complexities of large-sized multiple resources. Thus, this paper presents the development and application of the Elitist GA using the parallel scheme to find the optimal and/or near optimal solutions to the large-sized multiple RCPSP, as opposed to the existing studies that only considered small number of activities in construction engineering. The following sections briefly introduce the problem definitions for the RCPSP, followed by the development of the Elitist GA for solving the RCPSP. Experimental results on the standard problem sets are presented to show how the Elitist GA is competitive to the existing optimal and/or near-optimal solution methods under the same conditions.

2. Problem Description and Objective Function

The RCPSP has gained importance from the fact that it is a combinatorial problem, which includes job-shop sequencing and assembly-line balancing problems. This is well known as one type of a nondeterministic polynomial (NP)-hard problem (Blazewicz et al. 1983). A project that includes a finite set of activities is considered here, where N activities labeled i = 1, ···, n are given. Two major assumptions have been made for the implementation of the Elitist GA as follows: First, a project can be represented by a CPM network diagram where activities are numerically labeled such that successor activities always have higher numbers than all their predecessors. For every activity, the start and finish time are known. Second, modeling construction activities requires the use of material, labor, equipment, and costs. This research assumes that resources are to be of the renewable type and that the resource requirements are to be discrete. 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. The single-project multiple RCPSP can be formulated as follows:

$$\text{minimize } f(i) = \max\{t_i + d_i | i = 1,2,...,n\} \quad (1)$$

$$\text{subject to } t_j - t_i \geq d_i, \ \forall \ j \in S_i \quad (2)$$

$$\sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{i=1}^{N} H_{imt}^{RR} \leq \sum_{p=1}^{P}\sum_{m=1}^{M} \Xi_{mt}^{RR} \quad (3)$$

$$f_i \geq 0 \quad (4)$$

where, N is the total number of activities, P is the total number of time period, such as days, and M is the total number of resource types. The symbols of f(i), d_i, t_{ij}, and S_i stand for the fitness function of an individual, which means the finish time of activity i, the duration of activity i, the starting date of activity i and j, and the set of successors of activity i, respectively. H_{imt}^{RR} and Ξ_{mt}^{RR} means the resource requirement of mth resource of activity i on time p and the resource availability of mth resource of activity i on time p, respectively.

The fitness function is different from the objective function for the clarification of a computation process of the fitness value, that is, the project duration. It aims 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.

3. Elitist GA using Parallel Scheme

This section presents the development of the Elitist GA, which is an algorithm that incorporates the concept of elitism into the general genetic algorithm to find the optimal and/or near optimal solutions to the large-sized multiple RCPSP by preserving the best individual solution for the next generation so as to obtain the improved solution. The main procedure of the Elitist GA using the parallel scheme is shown in Figure 1. This research adopted a permutation-based encoding that was appropriate for solving the RCPSP (Hartmann 1998; Zhuang and Yassine 2004).

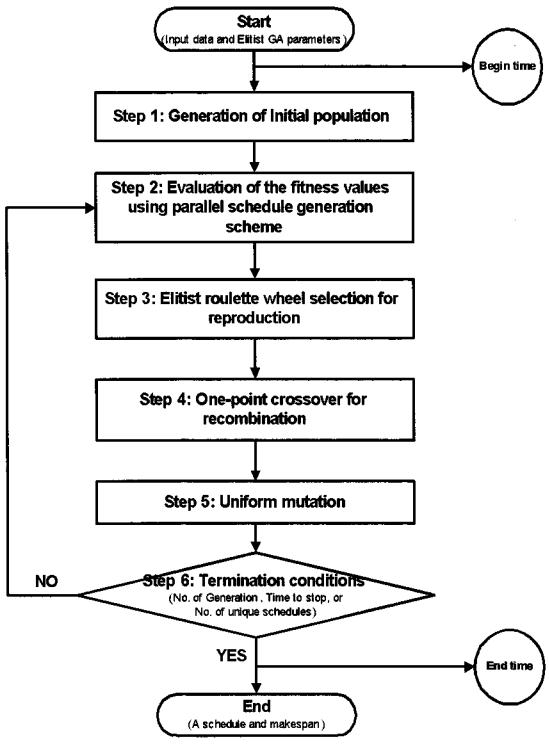


Figure 1. Flowchart for Elitist GA

For the encoding and decoding for the RCPSP, 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, as shown in Figure 2. 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.

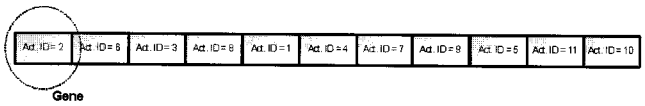


Figure 2. Chromosome (individual) representation

Step 1: Generation of initial population

An initial population composed of precedence feasible individuals was produced by the random number generator. An initial population is generated as follows:

(1) Select an activity from all unselected activity pool
(2) Check whether its immediate predecessor(s) are already selected
(3) Continue (2) until a satisfying activity is found if not yet selected
(4) Repeat (1) and (2) until the set of unselected activity is empty (As a result, an individual that consists of all activities is generated)
(5) Repeat (1) through (4) until all individuals of population size are generated

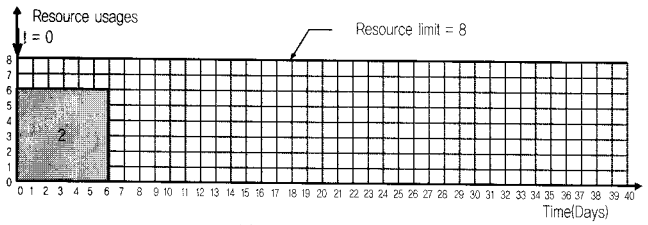
The random number generator simply provides precedence feasible solutions, but does not give the fitness value, a possible starting and finishing time of an activity, and the feasibility of resource constraints. It, for example, generates an individual {2, 7, 1, 6, 4, 3, 8, 9, 5, 11, 10} for 11 non-dummy activities. An individual cannot have duplicate activities as a result of checking precedence constraints among activities.

Step 2: Evaluation of the fitness values

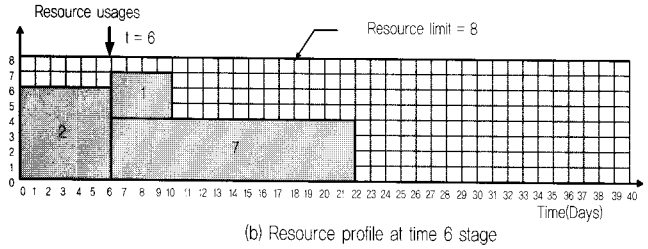
A schedule generation scheme is essential in most heuristic solution procedures for the RCPSP. It is a vital mechanism to compute a fitness value to the RCPSP. It is generally classified into two different mechanisms, depending on either as the activity increases (serial scheme) or as time increases (parallel scheme) in the heuristic approaches to the RCPSP. The parallel scheme, the algorithm of Kelley (1963) and the one of Brooks (Bedworth and Bailey 1982), was utilized in this paper to calculate the fitness value of an individual obtained from a population generated using the random number generator in Step 1. Its unique feature is that each stage n is associated with a schedule time tn, where $tm \leq tn$ for $m \leq n$ holds. On account of this schedule time, the set of scheduled activities is divided into the following two subsets: Activities which were scheduled and are completed up to the schedule time are in the complete set (Cn), while activities which were scheduled, but which are at the schedule time still active, are in the active set (An). The decision set (Dn) contains all yet unscheduled activities which are available for scheduling with regard to precedence and resource constraints. The partial schedule of each stage is made by the activities in the complete set and the active set. The schedule time of a stage is equal to the earliest completion time of activities in the active set of the previous stage.

Each stage is made up of two steps: First, the new schedule time is determined and activities with a finish time equal to the (new) schedule time are removed from the active set and put into the complete set. This, in turn, may place a number of activities into the decision set. Second, one activity from the decision set is selected according to the order of the activity list representation and scheduled to start at the current schedule time. Afterwards, this activity is removed from the decision set

and put into the active set. The second procedure is repeated until the decision set is empty, i.e., activities were scheduled or are no longer available for scheduling with regard to resource constraints. The parallel scheme terminates when all activities are in the complete set or active set. Take one potential activity list, {2, 7, 1, 6, 4, 3, 8, 9, 5, 11, 10} as an example for illustrating the mechanism applied in this paper. Figure 3 shows the resource profiles of the example at t=6 days.



(a) Resource profile at time 0 stage



(b) Resource profile at time 6 stage

Figure 3. Parallel schedule generation scheme

As shown in Figure 3 (a), the new schedule time is set to zero, as a starting point of scheduling the individual I= {2, 7, 1, 6, 4, 3, 8, 9, 5, 11, 10}. The first decision set {2, 1, 3} is created at time 0 stage because they are unscheduled and available for scheduling with regard to precedence relationships and resource constraints. All three activities, 2, 1, and 3, are considered to be scheduled at the same time. Activity 2 is selected first because it is placed in the first position of the decision set. Activity 2 then starts at time 0 for the duration of 6 because it does not have any predecessor and because its resource requirements (RR2=6) do not exceed the resource availability (RA=8, which is constant throughout the project duration) through the period of its duration. Simultaneously, Activities 1 and 3 are considered to be scheduled at time 0, but they cannot start because of the limitation of available resources.

Next, the new schedule time is determined at time 6 based on the earliest completion time (Day 6) of activity in the active set {2} of the previous stage, as shown in Figure 3 (b). At time 6 stage, the complete set {2}, and the new decision set {7, 1, 6, 3} are generated. Activities 1 and 3 are contained in the previous decision set, while Activities 7 and 6 are newly included from the list of the individual considered because their predecessor, Activity 2, is completed. In other words, the new decision set should be ordered into 7, 1, 6, and 3, but not 1, 3, 7, and 6 because the order of the activity list in the individual is {2, 7, 1, 6, 4, 3, 8, 9, 5, 11, 10}. Activities 7 and 1 are scheduled at time 7 for the duration of 16 and 4, respectively, because their resource requirements (RR7=4 and RR1+RR7=3+4=7) do not exceed the resource availability through the period of their durations. However, the rest of the decision set cannot be scheduled due to the limitation of available resources. The parallel scheme continues up to the last activity in the individual, as the time increases.

Step 3: Elitist roulette wheel selection for reproduction

The elitist preserving selection called elitism (De Jong 1975) was adopted to combine with the roulette wheel selection operator. Elitism roulette wheel selection operator was created using the pseudo-code as shown in Figure 4.

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 4. Pseudo-code of elitist roulette selection operator

Elitism 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 proposed by Holland

(1975) has been employed, as used in several studies (Hartmann 1998; Leu and Yang 1999). The concept of the selection was to determine selection probability for each individual proportional to the fitness value.

Step 4: One-point crossover for recombination

The goal of a crossover operator is to combine pieces of information coming from different individuals in the population. Preserving good building blocks and maintaining randomness and population diversity for searching non-redundant solutions is important. 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. 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 RCPSP. They were developed to deal with this type of ordering problem that occurred due to crossover operation.

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 (Goldberg 1989). 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 (Hartmann 1998; Zhuang and Yassine 2004). The theorem was proven by Hartmann (1998). The one-point crossover is operated as follows:

(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

The proportion of parent individuals performing the one-point crossover operator during a generation is controlled by the crossover probability , which determines how frequently the one-point crossover operator is invoked.

Step 5: Uniform mutation

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 serves as a means to avoid local minima in the search space. This operator can be ineffective because the genes in neighboring individual positions could be switched while still representing the same schedule. Therefore, a mutation on an individual does not necessarily change the related schedule because interchanging two activities, which have the same start time in the job sequence, is likely to change the individual, but not the related schedule. The uniform mutation operator is operated as follows:

(1) For each individual from a generation
(2) Generate a real random number
(3) Swap an activity after pivot point with activity at pivot point if a random number is equal to or less than mutation probability

Individuals are selected for the uniform mutation based on user defined mutation probability, which provides an expected value for the number of mutations to occur per generation.

Step 6: Termination conditions

Three different types of termination conditions can be determined to stop the run of the Elitist GA. They include the number of generations, timeout, and the number of unique schedules.

4. Experimental Results and Analysis

The Elitist GA 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 and mutation rates. Two output options, full and summary output, are available for data analysis. This section presents three computational experiment results.

4.1 Comparison with Heuristic and Other GA

First, a construction project schedule, which was extracted from the work of Shanmuganayagam (1989), was used to demonstrate the robustness of the Elitist GA by comparing the results produced from the Elitist GA with those obtained from the existing methods. Table 1 shows the schedule information.

Table 1. Schedules Informaiton for Case Example

Act.	Dur. (Days)	Depends on	Resource requirements R1 (Labor)	R2 (Equipment)	R3 (Equipment)
A	4	-	3	0	1
B	6	-	6	1	0
C	2	-	4	0	1
D	8	A	2	1	0
E	4	D	4	1	0
F	10	B	2	1	0
G	16	B	4	0	0
H	8	F	2	0	1
I	6	C	4	1	0
J	6	E, H	5	0	1
K	10	G, I	2	0	1
Maximum availability			8	1	1

To make an impartial comparison with the results obtained from the work of Chan et al. (1996), all activities were scheduled using just one resource (Labor, R1) with a fixed resource profile. The population size was set to 50 and the total number of generation was set to 40 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 2 shows the various schedule results in comparison to the single schedule by the heuristic rule (Shanmuganayagam 1989) and three schedules produced by GA-scheduler (Chan et al. 1996).

Table 2. Comparison of Various Schedules by Method

Activity No.	Res. Req.	Starting times of activities obtained by: Heuristic method	GA-scheduler (Chan et al. 1996)				Elitist GA (this research)		
		S1	S1	S2	S3	Elitist[1)]	S1	S2	S3
A	3	6	6	7	8	6	6	6	6
B	6	0	0	0	0	0	0	0	0
C	4	10	10	11	14	6	10	20	6
D	2	10	15	15	19	14	12	10	10
E	4	28	28	28	28	26	28	28	26
F	2	6	6	7	8	14	12	10	8
G	4	6	6	6	6	10	6	6	10
H	2	16	17	18	18	24	22	22	24
I	4	32	32	32	32	8	22	22	18
J	5	22	22	22	22	32	32	32	32
K	2	28	28	28	28	26	28	28	26

The Elitist GA produced the project duration of 38 days, which is same as those obtained either by the heuristic rule or by the GA-scheduler. It also generated 1,426 unique schedules, which amounts to 71.30% of the total schedules of 2000. A uniquely determined schedule (phenotype) computed using the parallel scheme can be related more than one individual (genotype), which 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. It took the total CPU time of 610 milliseconds for the algorithm to solve the RCPSP with single resource. Elitist GA ensures that small populations do not lose the current best solution in the population. The Elitist GA can provide several equally good and feasible scheduling alternatives, which indicate the similar result to GA-scheduler.

1) Obtained from the top of the individuals in the population of the last generation

4.2 Scheduling Project with Multiple Resources

To take into account of the multiple resources, secondly, the Elitist GA was run on the same example as used in Section 4.1. Labor (R1) and two types of equipment (R2 and R3) 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 for R1, R2, and R3, respectively. The population size, crossover and mutation rates were set to 50, 0.5, and 0.03, respectively, as evaluated and recommended by Goldberg (1989). The overall fitness value was obtained for multiple resources using the parallel scheme. Table 3 shows the partial schedules for multiple resources. The project duration was 54 days, which was obtained according to the partial schedules. The major differences between the schedule with single resource and the one with multiple resources are the expanded project duration and the different starting time of activities.

Table 3. Partial Schedules for Multiple Resources

Type		Parallel schedule generation scheme			
Stage No.	Time (Day)	Complete set (Cn)	Fitness Value (Day)	Decision set (Dn)	Active set (An)
1	0	{ - }	0	{2, 1, 3}	{2}
2	6	{2}	6	{7, 1, 6, 3}	{7, 1}
3	10	{2, 1}	10	{6, 4, 3}	{7, 6}
4	20	{2, 1, 6}	20	{4, 3, 8}	{7, 4, 8}
5	22	{2, 7, 1, 6 }	22	{3}	{4, 8}
6	28	{2, 7, 1, 6, 4, 8}	28	{3, 5}	{3, 5}
7	30	{2, 7, 1, 6, 4, 3, 8}	30	{9}	{5}
8	32	{2, 7, 1, 6, 4, 3, 8, 5}	32	{9, 11}	{9}
9	38	{2, 7, 1, 6, 4, 3, 8, 9, 5}	38	{11, 10}	{11}
10	48	{2, 7, 1, 6, 4, 3, 8, 9, 5, 11}	48	{10}	{10}
11	54	{2, 7, 1, 6, 4, 3, 8, 9, 5, 11, 10}	54	{ - }	{ - }

4.3 Comparison with Optimal Solutions

Finally, another experiment was conducted to verify the performance of the Elitist GA by comparing the results generated from the Elitist GA to those available from the well known optimal and/or lower bound solutions. Nine large-sized multiple RCPSP were obtained from the PSPLIB (Kolisch and Sprecher 1996) for this experiment. The problems consist of 30, 60, and 120 non-dummy activities, respectively. Each problem instance has four renewable resources. The overall performance of the Elitist GA was measured by the means of finding the best fitness value, which can be considered an optimal and/or near-optimal solution to the RCPSP. The input parameter values for the algorithm were set as follows: Initial population size, crossover and mutation rates were set to 50, 0.5, and 0.03, respectively. The termination condition was set to the maximum number of generations of 100, which brings 5,000 (50x100) trials.

Table 4 shows the minimum fitness values, total algorithm runtime in millisecond (ms), and the number of unique schedules as a result of scheduling nine different large-sized problems with multiple resources. Figure 5 shows the convergence behaviour for one of three 60-Activity projects. The Elitist GA found the optimal solutions for three 30-Activity projects and three 60-Activity projects, respectively, as they converges to a single point across the number of generation. It found the solutions very close to the lower bound solutions for three 120-Activity projects, even though it could not generate

Table 4. Comparison Results with Optimal Solutions

Project Group	Pro. ID	Minimum fitness (Project duration)	Total algorithm runtime (ms)	No. of unique schedules	Optimum /Lower bound solution[2]
30-Act.	1	43	2656	7040	43
	2	62	2719	7041	62
	3	38	2265	6983	38
60-Act.	1	65	3750	6948	65
	2	78	3828	7065	78
	3	60	3453	6996	60
120-Act.	1	114	10500	7000	99
	2	102	8578	7043	88
	3	97	8500	7023	84

2) The optimal solutions for the problems with 30 non-dummy activities used are known (Demeulemeester and Herroelen 1997), while for the problem instances with 60 and 120 non-dummy activities, only heuristic solutions, which are lower bound solution (Klein and Scholl 1999 for 60-Activity and Brucker and Knust 2003 for 120-Activity), are known.

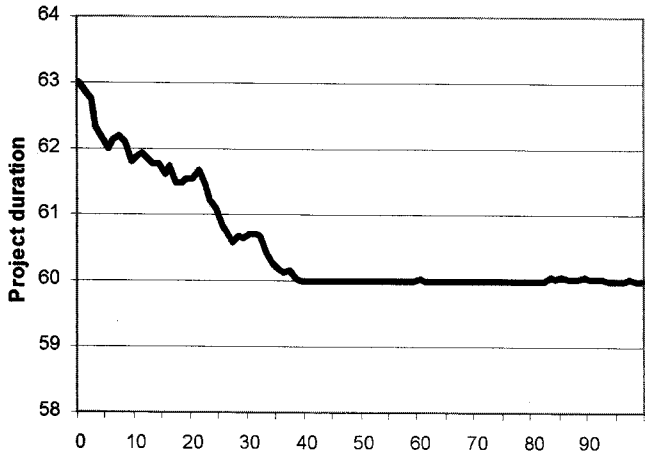


Figure 5. Convergence of Elitist GA

the identical project duration. The algorithm required more time to solve a larger problem than a smaller one as expected. It also generated similar number of unique schedules for each project group.

5. Discussions and Limitations

The standard RCPSP considered in this paper is of great practical importance because its general model can be used for application in a wide variety of scheduling applications. However, the problem needs to be modified or adjusted to model real construction site conditions. Some of the examples for modeling real situation using the Elitist GA are as follows: How to handle if available resources levels change over time, what if resources, i.e., crews can share jobs, what if the resources used are non-renewable, what if the resources are equipment-oriented, such as a crane, versus crews made of tradesman, how can resources be partially applied to more than one activity or used at a reduced rate, and how to address the different productivity of resources.

For the improvement of the Elitist GA application, two point or other crossover methods may also be employed on a random basis to improve the integrity of building blocks, even though the Elitist GA used only one-point crossover to apply the recombination theories fundamental to the GA. Although the Elitist GA eliminates infeasible schedules at the stage of random number generator, a hybrid-heuristic method can be used to assist the algorithm in examining potentially useful mutations and local minima problem. Finally, 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 that 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 GA since the selection pressure of pair-wise tournament selection is consistent regardless of the contents of the population.

6. Conclusions and Future Studies

This paper presented the development and application of an Elitist Genetic Algorithm using the parallel schedule generation scheme for solving a large-sized multiple RCPSP. Compared with a heuristic and other GA method, the Elitist GA provides one elitist individual schedule and several equally good scheduling alternatives for efficient decision making. The Elitist GA found the optimal solutions up to 60 non-dummy activity for each of six standard problems, while it found the solutions very close to the lower bound solutions for three 120 non-dummy activity projects. The Elitist GA also shows the capability to solve a large-sized multiple RCPSP by finding an optimal and/or near optimal solution within a reasonable amount of time. Thus, the Elitist GA can be an efficient algorithm that can apply to solve the large-sized multiple RCPSP.

This paper will help researchers and/or practitioners in the construction project scheduling software area with alternative means to find the optimal schedules by utilizing the advantages of the Elitist GA. More study is being conducted to find optimal solutions by incorporating a local search algorithm into the Elitist GA. In addition, a user friendly interface of the Elitist GA is under development for running numerous RCPSP at the same time.

References

1. Alcaraz, J. and Maroto, C. (2001). "A Robust Genetic Algorithm for Resource Allocation in Project Scheduling." Annals of Operations Research, 102, pp. 83-109.
2. Bedworth, D. D. and Bailey, J. E. (1982). Integrated Production Control Systems-Management, Analysis, Design, Wiley, New York. N.Y.
3. Blazewicz, J., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1983). "Scheduling Subject to Resource Constraint: Classification and Complexity." Discrete Applied Mathematics, 5, pp. 11-24.
4. Brucker, P. and Knust, S. (2003). "Lower Bounds for Resource-Constrained Project Scheduling Problems." European Journal of Operations Research, 149, pp. 302-313.
5. Brucker, P., Knust, S., Schoo, A., and Thiele, O. (1998). "A Branch and Bound Algorithm for the Resource-Constrained Project Scheduling Problems." European Journal of Operations Research, 107(2), pp. 272-288.
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), pp. 125-132.
7. 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.
8. Demeulemeester, E. L. and Herroelen, W. S. (1997). "New Benchmark Results for the Resource Constrained Project Scheduling Problem." Management Science, 43(11), pp. 1485-1492.
9. Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.
10. Hartmann, S. (1998). "A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling." Naval Research Logistics, 45, pp. 733-750.
11. Hartmann, S. (2002). "A Self-Adapting Genetic Algorithm for Project Scheduling under Resource Constraints." Naval Research Logistics, 49, pp. 433-448.
12 Hegazy, T. (1999). "Optimization of Resource Allocation and Leveling Using Genetic Algorithms." Journal of Construction Engineering and Management, ASCE, 125(3), pp. 167-175.
13. Hegazy, T. and Kassab, M. (2003). "Resource Optimization Using Combined Simulation and Genetic Algorithms." Journal of Construction Engineering and Management, ASCE, 129(6), pp. 698-705.
14. Hindi, K. S., Yang, H., and Fleszar, K. (2002). "An Evolutionary Algorithm for Resource-Constrained Project Scheduling." IEEE Transactions on Evolutionary Computation, 6(5), pp. 512-518.
15. Holland, J. K. (1975). Adaptation in Neural and Artificial Systems, University of Michigan Press, Ann Arbor, MI.
16. 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, pp. 347-365.
17. 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, pp. 322-346.
18. Kohlmorgen, U., Schmeck, H., and Haase, K. (1999). "Experiences with Fine-Grained Parallel Genetic Algorithms." Annals of Operations Research, 90, pp. 203-219.
19. Kolisch, R. and Sprecher, A. (1996). "PSPLIB - A Project Scheduling Problem Library." European Journal of Operational Research, 96, pp. 205-216.
20. Lee, J.-K. and Kim, Y.-D. (1996). "Search Heuristics for Resource-Constrained Project Scheduling." The Journal of the Operational Research Society, 47(5), pp. 678-689.

21. Leu, S. and Yang, C. (1999). "GA-Based Multicriteria Optimal Model for Construction Scheduling." Journal of Construction Engineering and Management, ASCE, 125(6), pp. 420-427.
22. Leu, S., Chen, A., and Yang, C. (1999). "Fuzzy Optimal Model for Resource-Constrained Construction Scheduling." Journal of Computing in Civil Engineering, 13(3), pp. 207-216.
23. Moselhi, O. and Lorterapong, P. (1993). "Least Impact Algorithm for Resource Allocation." Canadian Journal of Civil Engineering, CSCE, 20(2), pp. 180-188.
24. Shanmuganayagam, V. (1989). "Current Float Techniques for Resource Scheduling." Journal of Construction Engineering and Management, ASCE, 115(3), pp. 401-411.
25. Toklu, Y. C. (2002). "Application of Genetic Algorithms to Construction Scheduling With or Without Resource Constraints." Canadian Journal of Civil Engineering, 29, pp. 421-429.
26. Valls, V., Ballestin, F., and Quintanilla, S. (2005). "Justification and RCPSP: A Technique That Pays." European Journal of Operational Research, 165, pp. 375-386.
27. 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, pp. 1-11.

논문제출일: 2007.06.29
심사완료일: 2007.08.31