IEMS Vol. 8, No. 3, pp. 171-180, September 2009.

A Looping Population Learning Algorithm for the Makespan/Resource Trade-offs Project Scheduling

Ying-Chieh Fang
Department of Industrial Engineering and Management
Yuan-Ze University, 135 Yuan-Tung Road, Chung-Li, 320, Taiwan, R.O.C.
Tel: +886-3-463-8800, E-mail: iehshsu@saturn.yzu.edu.tw

Chiuh-Cheng Chyu†
Department of Industrial Engineering and Management
Yuan-Ze University, 135 Yuan-Tung Road, Chung-Li, 320, Taiwan, R.O.C.
Tel: +886-3-463-8800, E-mail: iehshsu@saturn.yzu.edu.tw

Received Date, March 25, 2009; Revised Date, July 31, 2009; Accepted Date, August 3, 2009 (Selected from APIEMS 2008)

Abstract. Population learning algorithm (PLA) is a population-based method that was inspired by the similarities to the phenomenon of social education process in which a diminishing number of individuals enter an increasing number of learning stages. The study aims to develop a framework that repeatedly applying the PLA to solve the discrete resource constrained project scheduling problem with two objectives: minimizing project makespan and renewable resource availability, which are two most common concerns of management when a project is being executed. The PLA looping framework will provide a number of near Pareto optimal schedules for the management to make a choice. Different improvement schemes and learning procedures are applied at different stages of the process. The process gradually becomes more and more sophisticated and time consuming as there are less and less individuals to be taught. An experiment with ProGen generated instances was conducted, and the results demonstrated that the looping framework using PLA outperforms those using genetic local search, particle swarm optimization with local search, scatter search, as well as biased sampling multi-pass algorithm, in terms of several performance measures of proximity. However, the diversity using spread metric does not reveal any significant difference between these five looping algorithms.

Keywords: Project Scheduling, Population Learning Algorithms, Genetic Algorithms, Particle Swarm Optimization, Scatter Search, Forward/Backward Improvement

1. INTRODUCTION

The objective of a project that project management plans to achieve should be clear and measurable. Generally, there are three fundamental objectives of the project management: time, cost and quality (Demeulemeester and Herroelen, 2002). The relative importance of each objective will differ for different projects. In many cases, especially for product innovation and new product development projects, time is often considered to be the first priority among performance objectives. However, in other cases such as plant construction and software system projects, resource availability, normally referring to renewable resources, may become the most concern of the project management because it comprises the major cost of the project. Some examples of renewable resources are equipments, technicians, etc. The project quality is usually measured by the degree to which a project's outcome conforms to the customer's requirements and the degree to which the project completes within budget and on schedule (Erenguc and Icmeli-Tukel, 1999).

Over the past few decades, a great deal of research effort in project scheduling has been devoted to developing exact and heuristic procedures for the resource constrained project scheduling problem (RCPSP) with the aim of minimizing makespan. The RCPSP specifies constant resource availability for each resource type throughout the project execution, as well as resource consumption per period during the execution of each activity. Such a problem is often referred to as resource constrained project scheduling problem (RCPSP) with the objective of minimizing the project makespan. Blazewicz *et al*. (1983) has shown that this optimization problem is NP-hard in the strong sense.

† : Corresponding Author

Many solution approaches have been proposed to solve the RCPSP with makespan minimization objective, including branch and bound algorithm (B&B), biased sampling with local search (LS), genetic algorithm (GA), ant colony optimization (ACO), and hybrid metaheuristic approaches, such as scatter search/electromagnetism (Debels *et al*., 2006). Kolisch and Hartmann (2006) provided a valuable survey and investigation on algorithms for this problem. The survey indicates that hybrid approach with the forward/backward improvement (FBI; Li and Willis 1992, Valls *et al*., 2005) method is one of the most efficient and effective solution approaches to this problem. Tormos and Lova (2003) presented a hybrid multi-pass method that combines a biased random sampling procedure with a FBI method.

On the other hand, compared to the minimizing makespan problem, relative few studies have been focused on the RCPSP with the objective of minimizing resource availability. Yamashita *et al*. (2006) proposed a scatter search algorithm for solving the SRCPSP with the objective of minimizing total resource availability cost per period, subject to a project deadline. Hsu and Kim (2005) introduced a priority rule heuristic for a multi-mode resource investment problem. The problem attempts to find precedence, renewable resource- and deadline-feasible schedule with the minimum resource investment cost per period. The cost of each resource type per period is unit cost times the amount available per period, and the resource investment cost per period is defined as the total cost given the availability setting for all resource types.

Recently, considerable research efforts have been spent on RCPSP considering the duration/cost tradeoff relations for each activity of the project. Throughout this paper, we shall refer to duration as the processing time of an activity, and makespan as the completion time of a project. Erenguc *et al*. (2001) brought forward a branch and bound method for the RCPSP with multiple crashable modes, in which the duration/cost of an activity is determined by the mode selection and the duration (crashing) within the mode. Vanhouche *et al*. (2002) focused on the discrete duration/cost trade-offs problem in project scheduling with time-switch constraints, and proposed a branch and bound algorithm, as well as a heuristic procedure, to solve this particular problem. The problems of continuous duration/cost trade-off with linear, convex, and concave cost-duration functions had been discussed in Demeulemeester and Herroelen (2002). These problems did not take resource constraints into consideration, and the piecewise-linear approximation solution method was introduced. A discrete-continuous duration/cost trade-off problem with makespan minimization objective was discussed by Józefowska *et al*. (1998), where the duration of each activity is a function of the single continuous resource. Waligóra (2008) presented a tabu search method for solving two payment models of the same trade-off problem with the objective of maximizing the net present value.

The problem under study is an RCPSP that simultaneously considers two objectives: project makespan and resource availability. Using the classification scheme by Herroelen *et al*. (1999), the problem can be denoted as $m,1/cpm/(C_{max}, av)$. Unlike the duration/cost tradeoffs problem mentioned previously, we consider the tradeoffs relations of the project as a whole, instead of each respective activity. The focus of the study is to develop a looping population learning algorithm (PLA) capable of providing a set of near Pareto optimal feasible project schedules within an acceptable time. Several other looping hybrid algorithms are used for comparison purposes.

The paper is structured as follows. In section 2, we describe the discrete resource constrained makespan/resource tradeoffs project scheduling problem. Section 3 illustrates several algorithms for the problem under study, including the PLA. Section 4 presents our numerical results, using two sets of instances from PSPLIB (Kolisch and Sprecher 1996; http//129.187.106.231/psplib/). The last section concludes this paper and suggests a further research direction.

2. PROBLEM DESCRIPTIONS

This paper studies an RCPSP considering the tradeoffs between the project makespan and resource availability. Many research articles focus on minimizing one of these two objectives (Demeulemeester and Herroelen, 2002; Kolisch and Hartmann, 2006; Yamashita *et al*., 2006).

The resource-constrained project scheduling problem can be stated as follows. A project consists of a set of activities (nodes), $N = \{0, 1, \cdots, J+1\}$, where activities 0 and $J+1$are dummy and represent the events of the start time and the completion time of the project, respectively. A set of precedence relationships (arcs) between pairs of activities must be specified in the project. These activities and their associated precedence relationships can be represented by an acyclic directed network $G = (N, A)$, where A is the set of arcs.

In the study, each activity has to be processed without interruption to complete the project (nonpreemption). The duration of an activity j is denoted by d_j, and its request amount for resource type k is r_{jk}. Both d_j and r_{jk} are predefined and cannot be changed over the whole project duration. For the dummy activities, $d_0 = d_{J+1} = 0$, and $r_{0k} = r_{J+1,k} = 0$, for any resource type k. The availability of each resource type k in each time period is R^k units, $k = 1, \cdots, K$. A project schedule S can be represented by the start time of each activity, $(S_0, S_1, \cdots, S_{J+1})$ with $S_0 = 0$ and project makespan S_{J+1}, or the finish time of each activity, $(f_0, f_1, \cdots, f_{J+1})$ with $f_0 = 0$ and $f_{J+1} = S_{J+1}$. Note that $f_j = S_j + d_j$ for all j. A schedule S is feasible if it satisfies both precedence and resource constraints at any execution time period. The objective of

the RCPSP is to construct a near Pareto front for the pair of objectives: minimizing (a) project makespan and (b) total resource availability. The mathematical formulation of the problem that refers to Herroelen *et al*. (1999) is shown as follows.

$$Minimize\ (S_{J+1},\ \sum_{k=1}^{K} R^k)$$

Subject to

$$S_i + d_i \le S_j \quad \text{for all arcs} < i,\ j > \in A \quad i = 0,\ \cdots, J+1$$

$$\sum_{i \in S(t)} r_{ik} \le R^k \quad \text{for} \quad k = 1,\ \cdots, K \qquad t = 1,\ \cdots, T$$

where $S(t)$ is the set of activities in progress during time period $[t\text{-}1,\ t]$ and T is an upper bound on the project makespan.

Figure 1 displays a project example with one resource type and eight activities, which are represented by nodes 1 to 8. Node 0 denotes the starting event whereas node 9 the ending event. Both are dummy nodes with zero duration and resource usage. In this example, $r_{11} = 2$, $d_1 = 3$ and we will assume that $R^1 = 4$.

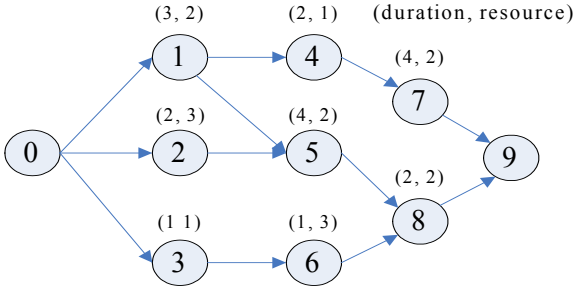

Figure 1. A project with one resource type.

3. PROBLEM SOLVING METHODS

In an acyclic directed network, there exists a topological ordering on nodes such that no node is numbered smaller than any of its predecessors. An activity list (AL) is a sequence of the activities that follows the precedence constraints. There are two methods to decode an AL as a project schedule: one is serial schedule generation scheme (S-SGS), and the other is parallel schedule generation scheme (P-SGS). The former will produce an active schedule, whereas the latter will produce a non-delay schedule. It is well known that optimal schedules must be active schedules, which include non-delay schedules. In our algorithm, S-SGS is used for decoding AL.

In order to construct a near Pareto front based on the two objectives mentioned in section 2, the paper proposes the framework as shown in Algorithm Main Framework, in which an algorithm for solving the RCPSP with the objective of minimizing makespan can be repeatedly employed.

Algorithm Main Framework:

Step 1: Employ the critical path method (CPM) to find the minimum makespan T_{CPM}, as well as the quantity consumed of each resource type corresponding to the CPM schedule, ($R^1_{\max}, \cdots, R^K_{\max}$). Compute the maximum total quantity of resource from the CPM schedule: MaxRsum = $R^1_{\max} + \cdots + R^K_{\max}$, as well as the minimum total quantity of resource required for executing every activity, MinRsum = $R^1_{\min} + \cdots + R^K_{\min}$, where $R^k_{\min}$ is the minimum quantity required of resource type k to implement all activities, i.e., $R^k_{\min} = \text{Max}\{r_{jk} \mid j = 1,\ \cdots, J\}$.

Step 2: Select randomly one of resource types from $\{1,\ \cdots, K\}$ and decrease its available amount by one unit. If this type has been consecutively chosen for four times, give up the choice and begin another selection. Whenever a resource type k has declined to its minimum amount $R^k_{\min}$, remove type k from the candidate list.

Step 3: Employ an efficient algorithm to find the minimum or a near minimum makespan of the RCPSP with the resource availability obtained in Step 2.

Step 4: If all resource types have reached its minimum required amount, go to Step 5; otherwise, go to Step 2.

Step 5: Construct the near Pareto front by comparing all the best solutions obtained.

In Step 3, five algorithms are applied to solve the RCPSP with minimum makespan: (1) Combining multiple pass method (Boctor, 1990) with parameterized regret-based biased random sampling (Kolisch and Drexl, 1996), denoted as (MP-RBRS); (2) Genetic local search (GLS); (3) Particle swarm optimization with local search (PSO-LS); (4) Scatter search (SS); (5) Population learning algorithm (PLA). All five algorithms are similar in structure to those that have been used in the literature. The PLA was originally introduced by Jedrzejowicz (1999). Based on the PLA concept, we have devised a PLA covering the first four aforementioned algorithms in the evolutionary stages.

In the framework, whenever the total quantity of resource is reduced by one unit, the selected algorithm computes five thousand solutions (in Step 3) to find a near-optimal makespan. The five algorithms are described below.

3.1 Multi-pass regret biased random sampling (MP-RBRS)

In this algorithm, five priority rules are used: shortest

process time (SPT), minimum slack time (MST), earliest start time (EST), earliest finish time (EFT), and latest finish time (LFT). At each step, one of the five rules is firstly chosen at random. Then the RBRS scheme based on the selected priority rule is applied to choose an activity from the candidate list. The same procedure is repeated until an AL has been attained. Afterwards, the serial schedule generation scheme (S-SGS) is employed to decode this AL to an active schedule, and then the FBI (forward/backward improvement) is applied to refine this schedule (see Figures 1 and 4~5 for examples). The MP-RBRS algorithm computes five thousand solutions and return with the best one. The following describes the RBRS procedure:

Let D be the candidate list of activities at the current step. The regret value (ρ_i) for each activity i in D compares the priority value of activity i, $v(i)$, with the worst priority value $v(j)$ of the activities of D and is calculated as follows:

$\rho_i = \max\{v(j) \mid j \in D\} - v(i)$, and the parameterised probability (ϕ_i) is calculated as follows:

$$\phi_i = \frac{(\rho_i + \varepsilon)}{\sum_{j \in D}(\rho_j + \varepsilon)}, \quad (1)$$

where ε is a predetermined parameter value.

3.2 Genetic local search (GLS)

The GLS adopts an elitist strategy to maintain good quality solutions in each generation. In this algorithm, roulette wheel is used for reproduction, one point crossover for producing offspring, two-swap operation for mutation, and the FBI for further improvement. Valls *et al*. (2005) and Kolisch, and Hartmann (2006) have shown that the FBI is the most effective local search in the project scheduling problem with makespan minimization objective. Our GLS also adopts the FBI to refine a new GA solution, but the means to perform one point crossover and two-swap mutation is different from those by Alcaraz *et al*. (2004) and Valls *et al*. (2005). Figure 2 illustrates our mutation using Figure 1. A mutation consists of four steps: (1) Convert an AL to a weight list (WL), where the weight of an activity is the list order of the activity in the AL; (2) Randomly select two positions in the WL and swap their weights; in the example, weights of activities 2 and 7 are swapped; (3) Convert the WL to AL based on precedence and weights. The one point crossover also applies the conversion process.

The algorithm terminates when five thousand solutions have been attained. The population size is set to 50, the crossover rate is 0.9, and the mutation rate is 0.1.

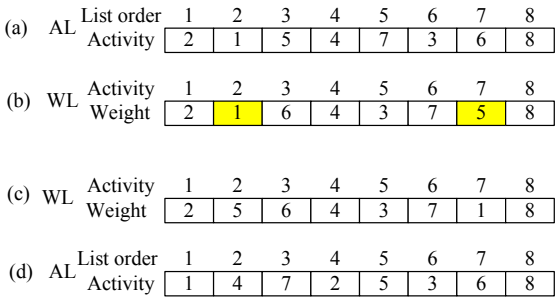

Figure 2. An example of mutation.

3.3 Particle swarm optimization with local search (PSO-LS)

Kennedy and Eberhart (1995) proposed PSO, which was inspired by the choreography of a bird flock. The idea of this approach is to simulate the movement of a group (or population) of birds which aim to find food. This approach can be viewed as a distributed behavioral algorithm that performs multidirectional search. In the proposed algorithm, a particle $P[i]$ is a random key list (RKL), which is a representation of a project schedule. An RKL can be converted to an AL and then the AL is decoded as an active project schedule through the S-SGS. Figures 3 to 6 depict the evaluation process of a particle $P[i]$ using the Figure 1 example. Figure 3 shows the converting process from an RKL to an AL. Figure 4 presents the project schedule corresponding to the AL via S-SGS. Figure 5 shows the FBI procedure. Figure 6 presents a better quality RKL which is obtained by reordering the random key numbers according to the starting times of the FBI schedule in increasing order.

Our PSO is different in that it applies the FBI local search. Zhang *et al*. (2005) employed PSO to solve RC PSP using continuous and discrete solution representation schemes, particle-representation priority and particle-representation permutation. The two schemes function similarly to RKL and AL. The population size adopted is equal to the number of non-dummy activities in the project. Kemmoé Tchomté and Gourgand (2009) proposed a particle displacement PSO and applied its discrete version to solve RCPSP using left- and right-shift local search.

For PSO with continuous representation scheme, the acceleration coefficients, c_1 and c_2, are typically set to a value of 2.0 (Eberhart *et al*., 1996). Suganthan (1999) has shown that setting $c_1 \neq c_2$ can lead to improved performance. In our PSO, we tested c_1 and c_2 near 2.0, and obtained $c_1 = 2.0$ and $c_2 = 2.1$. The population size is set to a constant of 30 for the benchmark instances, which consist of 30 and 60 activities. The inertia weight, w, is set to 0.3 based on several test results.

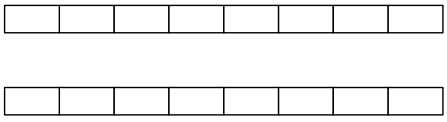

Figure 3. Converting RKL to AL.

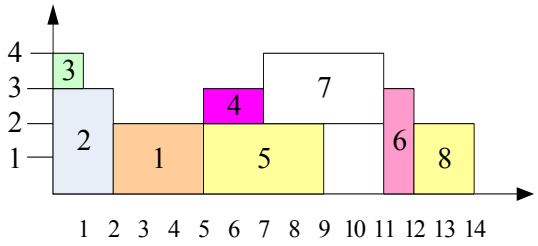

Figure 4. Project schedule corresponding to AL in Fig. 2.

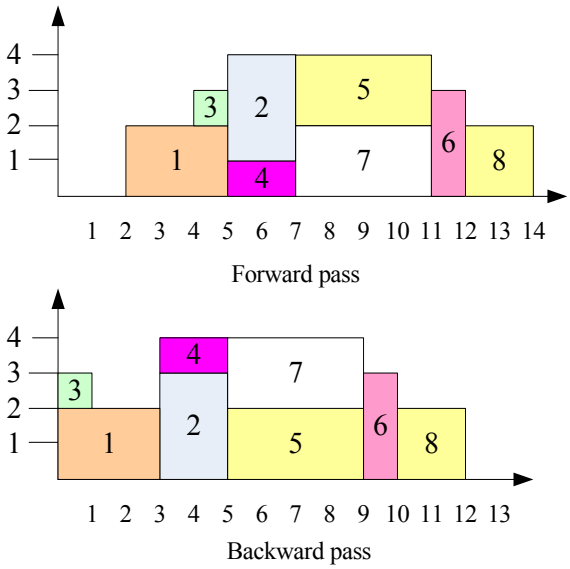

Figure 5. FBI procedure.

Random key list	1	2	3	4	5	6	7	8
	1.11	2.56	2.43	2.95	3.68	4.33	3.72	6.23

Figure 6. RKL of the schedule after FBI.

PSO-LS algorithm

Step 0: Set population size M = 30; initialize the speed of particle i, $V[i] = 0$; initial weight w = 0.3; two constants $c_1 = 2$, $c_2 = 2.1$;

Step 1: For $i = 1$ to M
Randomly generate a particle $P[i]$;
Evaluate $P[i]$ (apply converting process to obtain a schedule);
End for
GLbest = PObest = Best{$P[i]$: $i = 1, \cdots, M$}

Step 2: Repeat the following procedure until 5000 solutions have been attained.
Begin
For $i = 1$ to M
$V[i] = w \cdot V[i] + c_1 \cdot B_1 \cdot (\text{PObest} - P[i]) + c_2 \cdot B_2 \cdot (\text{GLbest} - P[i])$;
(Calculate speed of each particle, w = Inertial weight, B_1 and B_2 are random numbers in [0, 1])
$P[i] = P[i] + V[i]$; Evaluate $P[i]$;
If $P[i]$ is better than PObest, then PObest = $P[i]$;
If PObest is better than GLbest,
then GLbest = PObest;
End For
End (Begin).

3.4 Scatter search

Scatter search (SS), proposed by Glover (1977, 2003), is a population-based heuristic that has shown great potential for solving many hard combinatorial optimization problems, such as RCPSP and its variants (Debel *et al.* 2006; Yamashita *et al.*, 2006), and capacitated facility location problems (Keskin and Üster, 2007). The fundamental concepts and principles are presented in Martı' *et al.* (2006). We develop a looping framework that uses scatter search with hybrid improvements, including FBI local search and path-relinking routines for the make/resource trade-offs project scheduling. Each scatter search iteration calculates five thousand solutions.

A scatter search consists of five components: diversification generation, improvement, reference -set update, solution recombination, and subset generation. In our algorithm, the reference set size is set to 10 and consists of two types of solutions: quality and diversity. We tested three combinations of solution types: 8:2, 7:3, and 6:4. The results indicate that the ratio, 7:3, is the most appropriate. For subset generation method, we consider all pair combinations, which are 45 pairs. To perform solution recombination for each pair, we adopt three-stage path-relinking. In each stage, we randomly choose one-fourth of positions, convert AL to WL, swap the weights, and convert WL back to AL. The best solution found in this stage is improved by FBI. Meanwhile, the resulting 45 solution pairs in each stage will become the parent solution pairs for the next stage. Furthermore, the distance between two solutions x and y is defined as the number of positions of different activities in the corresponding two ALs. The diversity metric of a solution x is defined as the smallest distance from the seven quality solutions in the reference set.

The scatter search used in the study is described as follows:

Step 1: Randomly generate 500 Activity lists. Select seven best quality solutions and three most diverse solutions into the reference set.

Step 2: Perform subset combination.

Step 3: For every pair combination, perform three-stage solution recombination; perform FBI for each pair of solutions at the end of last stage.

Step 4: If termination condition is not satisfied, update the reference set and go to Step 2; otherwise, terminate the algorithm.

3.5 Population learning algorithm (PLA)

PLA was originally introduced by Jedrzejowicz (1999). The PLA is a population-based method inspired by analogies to a phenomenon of social education process in which a diminishing number of individuals enter more and more advanced learning stages. It possesses the following characteristics: (1) A huge number of individuals enter the system, (2) Individuals learn through organized environment as well as self study, (3) Learning processes is inherently parallel (different learning environments), (4) Learning process is divided into stages, (5) More advanced stages are entered by a diminishing number of individuals from the initial population, (6) At higher stage more advanced learning and improvement techniques are used, and (7) A final stage is reached by only a fraction of the initial population.

The proposed PLA consists of an initial stage and three learning stages:

- In the initial stage, RBRS-MP is employed to generate a large number of diversified individuals. Select the best half of individuals in the population and enter the first stage.
- The learning process in the first stage adopts an elitist hybrid evolutionary algorithm (HEA). At each generation step, 70% of individuals are produced using single point crossover, 20% of individuals using 2-swap mutations, and 10% of individuals using the bidirectional planning heuristic (Kiein, 2000). The implementation of one bidirectional planning procedure requires taking two individuals from the parent population. All offspring produced in this generation and all individuals in the parent population will compete for the memberships of the next population. Select a fraction of individuals in the final population to enter the second stage.
- In the second stage, PSO-LS is used as an advanced learning tool. It should be noticed that before applying this algorithm, all individuals of the initial population coming from the second stage must be converted into random key representations.
- In the final stage, scatter search (SS; Glover and Kochenberge, 2003; Laguna and Glover, 2006; Yamashita and Armentano, 2006) is applied. The reference set consists of four high quality solutions and one diverse solution. The first reference set takes the best four solutions and the most diversified solution from the final population at the second stage. To find a most diversified solution, we define the distance measure between two individuals as

$$D(\mathrm{RKL}_i, \mathrm{RKL}_k) = \sum_{j=1}^{J} |\mathrm{RKL}_i[j] - \mathrm{RKL}_k[j]|,$$

and we define the diversity measure of an RKL as $Div = Min\{D(\mathrm{RKL}, \mathrm{RKL}_q) \mid q \in Q\}$, where Q is the set of quality solutions in the reference set. A most diversified solution in a group is the RKL that has the maximum value of Div.

The other elements of the SS used in this stage are described below.

(a) Two-element subset generation method. This method will generate a total of 10 pairs of RKLs. For each pair, the RKL with higher quality will serve as the initiating solution and the other as the guiding solution.

(b) Path re-linking solution recombination method. An iteration of solution recombination method produces four offspring. Let x and y be the initiating solution and guiding solution, respectively. To produce the first offspring, one-fourth of positions in x are randomly chosen and these random key numbers are replaced with those in the same positions in y. Denote the new RKL as x_1. The same procedure is applied to x_1 and y to produce the second offspring x_2, and so on. The best schedule among the four RKLs will be further refined by the FBI procedure, and the resulting solution will be converted back to RKL, which will be used to update the reference set.

Since all four algorithms are effective in solving the RCPSP, our order of matching the PLA algorithm in stages is primary based on the suitable population size for each algorithm. We observed in the literature that the rank of population sizes commonly adopted is HEA, PSO-LS, and SS. The RBRS-MP is suitable for generating a large number of diverse and quality solutions. In SS, the size of reference set is small due to the fact that a reference set will generate a considerable number of solutions in two-element subset generation procedure and path re-linking solution recombination procedure. These two procedures can be viewed as intensive local search or advanced learning procedure. In the PLA, the output solutions of each stage are screened and selected as the input for the next stage. The number of solutions distributed to the four stages are 100, 2000, 2000, and 900, respectively.

4. EXPERIMENTAL RESULTS

In this section, the performances of the proposed algorithms for the discrete makespan/cost tradeoffs RCPSP are evaluated and then compared against each other via two sets of benchmark instances, each of which was drawn from PSPLIB with resource strength (RS) equal to 0.5. The following performance metrics are used: (1) C measure for pair-wise comparison, (2) Error rate (ER), Generational distance (GD), and Overall non-dominated vector generation rate (ONVGR) for measurement of convergence or proximity, and (3) Spread for measurement of diversity. Set 1 contains four instances, each of which has 30 activities, four resource types, and network

complexity (NC) of 2.1; that is**,** each activity has an average of 2.1 successors. In addition, the resource factors (RF) of the four instances are 0.25, 0.5, 0.75, 1.0, which are denoted as j30_1 to j30_4, respectively. The RF refers to the fraction of resource types used by each activity in a project. For example, RF = 0.5 means that each activity consumes two resource types. Set 2 contains four instances of 60 activities with the same problem parameter values as in Set 1. All algorithms were coded in Borland C++Builder 6, and tested on a computer with Intel core dual, 1.86GHz and 2 Giga bytes DDR667. Table 1 displays the computation times of the algorithms on each instance. The SS based algorithm takes longer times than the others. Such a result is likely due to the update process for the diverse solutions in the reference set.

Table 1. Computation times of algorithms.

Computation times(in seconds)								
algorithm	j30_1	j30_2	j30_3	j30_4	j60_1	j60_2	j60_3	j60_4
PLA	7.63	8.94	10.25	10.31	36.74	39.64	42.54	45.25
GLS	7.05	7.36	7.69	7.38	31.13	31.81	31.86	32.53
PSO-LS	8.32	8.65	8.65	8.98	38.87	38.88	38.92	39.59
MP-RBRS	5.76	5.78	6.09	6.40	23.89	24.17	24.35	24.71
SS	8.77	10.73	11.79	12.51	40.78	44.79	55.72	57.12

In the absence of any known true Pareto-front, we computed reference set as non-dominated solutions from the combination of all the five solution sets produced by PLA, MP-RBRS, GLS, PSO-LS, and SS. The performances of these algorithms based on the metrics are presented as follows.

4.1 Pair-wise Comparison

C measure (Zitzler and Thiele, 1999) compares two sets directly and indicates coverage of one set over another. It does not require any reference points to compute the measure and hence, is not sensitive to any external input. Let A and B be the non-dominated solutions sets computed by algorithms P_A and P_B, respectively. The measure $C(P_A, P_B)$ maps the ordered pair (A, B) to the interval [0, 1], is given by

$$C(P_A, P_B) = \frac{|\, b \in B : \exists\, a \in A \ni a \preceq b \,|}{|B|}, \qquad (2)$$

where $a \preceq b$ means that solution a is not dominated by b. If $C(P_A, P_B)$ equals 1, it would imply that all representative solutions found by algorithm P_B are covered by those found by algorithm P_A. In other words, algorithm P_B does not produce any solution that is neither dominated nor the same as any solution produced by P_A. Table 2 displays the pair-wise comparison results for the eight instances. Apparently, the PLA outperforms the others in this measure, and the GLS ranks second when performance is viewed on all instances. On the other hand, the PSO underperforms the others. This may be due to the fact that the feature of PSO and the RKL is suitable for continuous objective functions. There are many RKLs corresponding to one AL, which will make solution search ineffective.

Table 2. *C measure* on the four algorithms(in %).

C (Prow, Pcolumn)	J30_1					J60_1				
	PLA	GLS	PSO-LS	MP-RBRS	SS	PLA	GLS	PSO-LS	MP-RBRS	SS
PLA	\	0.00	0.00	0.00	0.00	\	0.50	0.20	0.36	0.40
GLS	1.00	\	0.20	0.50	0.25	0.60	\	0.20	0.36	0.50
PSO-LS	1.00	0.50	\	1.00	0.50	0.80	0.83	\	0.64	0.60
MP-RBRS	0.67	0.25	0.00	\	0.25	0.60	0.75	0.40	\	0.30
SS	1.00	0.75	0.40	0.50	\	0.60	0.58	0.40	0.64	\
C measure	J30_2					J60_2				
	PLA	GLS	PSO-LS	MP-RBRS	SS	PLA	GLS	PSO-LS	MP-RBRS	SS
PLA	\	0.36	0.09	0.10	0.10	\	0.74	0.39	0.05	0.24
GLS	0.25	\	0.00	0.10	0.00	0.14	\	0.06	0.57	0.10
PSO-LS	0.88	1.00	\	0.50	0.50	0.59	0.90	\	0.86	0.38
MP-RBRS	0.75	0.73	0.27	\	0.30	0.90	0.39	0.06	\	0.10
SS	0.88	1.00	0.36	0.50	\	0.64	0.84	0.33	0.86	\
C measure	J30_3					J60_3				
	PLA	GLS	PSO-LS	MP-RBRS	SS	PLA	GLS	PSO-LS	MP-RBRS	SS
PLA	\	0.00	0.43	0.53	0.61	\	0.06	0.28	0.38	0.31
GLS	1.00	\	0.00	0.00	0.00	0.90	\	0.22	0.09	0.37
PSO-LS	0.56	1.00	\	0.33	0.22	0.61	0.60	\	0.35	0.57
MP-RBRS	0.39	0.95	0.48	\	0.28	0.50	0.83	0.53	\	0.60
SS	0.39	1.00	0.52	0.53	\	0.67	0.48	0.31	0.29	\
C measure	J30_4					J60_4				
	PLA	GLS	PSO-LS	MP-RBRS	SS	PLA	GLS	PSO-LS	MP-RBRS	SS
PLA	\	0.15	0.15	0.00	0.12	\	0.05	0.07	0.30	0.10
GLS	0.88	\	0.44	0.41	0.44	0.89	\	0.00	0.02	0.08
PSO-LS	0.88	0.64	\	0.44	0.69	0.88	1.00	\	0.81	0.51
MP-RBRS	1.00	0.56	0.48	\	0.63	0.63	0.95	0.20	\	0.13
SS	0.81	0.36	0.16	0.30	\	0.90	0.97	0.41	0.84	\

4.2 Convergence Measure

Three metrics are proposed to measure the closeness of a solution set to a Pareto set or a reference set.

The first metric is error ratio (ER), proposed by Veldhuizen (1999). It is defined as the ratio of the number of solutions not contained in the global reference set over the number of representative solutions found by the algorithm. The drawback of this measure is that it will not reflect a true performance when the reference set is not large and dense enough. The second metric is overall non-dominated vector generation ratio (ONVGR; Van Veldhuizen, 1999), which is similar to the first one, except that the denumerator is replaced with the number of solutions in the reference set. The third one is genera-

tional distance (GD; Deb, 2002), which is defined as follows:

$$\mathrm{GD}(A) = \frac{\sum_{i=1}^{|A|} h_i}{|A|}, \text{ where} \tag{3}$$

$$h_i = Min_{j=1}^{|P^*|} \sqrt{\sum_{k=1}^{2} \left(\frac{g_k^i - g_k^j}{g_k^{\max} - g_k^{\min}} \right)^2} \tag{4}$$

In the above mathematical expressions, A is the representative solution set found by a proposed algorithm, $g_k^{\max}$ and $g_k^{\min}$ are the maximum and minimum function values, respectively, of the kth objective function in the reference set P^*. Here, $g_2^{\max}$ is the sum of maximum resource usage and $g_1^{\min}$ is the makespan corresponding to the CPM schedule, respectively; $g_2^{\min}$ is the sum of minimum required units of each resource type; $g_1^{\max}$ is the best makespan of the minimum resource availability problem, found by the four proposed heuristics based on five solutions.

Table 3. Convergence performance of the algorithms.

	j30_1 (RF = 0.25)			j60_1 (RF = 0.25)		
algorithm	ER	ONVGR	GD	ER	ONVGR	GD
PLA	0.6	0.5	0.0427	0.3	0.7	0.00302
GLS	0.75	0.25	0.0264	0.917	0.1	0.0249
PSO-LS	1	0	0.0657	0.7	0.3	0.0187
MP-RBRS	1	0	0.0822	0.818	0.2	0.0193
SS	0.5	0.5	0.0196	0.6	0.4	0.0206
	j30_2 (RF = 0.50)			j60_2 (RF = 0.50)		
algorithm	ER	ONVGR	GD	ER	ONVGR	GD
PLA	0.273	0.615	0.00388	0.444	0.357	0.00491
GLS	1	0	0.0423	0.968	0.0357	0.0164
PSO-LS	0.875	0.0769	0.0281	0.682	0.25	0.00521
MP-RBRS	0.7	0.231	0.0103	0.905	0.0714	0.0152
SS	0.8	0.154	0.00817	0.476	0.393	0.00515
	j30_3 (RF = 0.75)			j60_3 (RF = 0.75)		
algorithm	ER	ONVGR	GD	ER	ONVGR	GD
PLA	0.667	0.273	0.00792	0	1	0
GLS	1	0	0.0344	1	0	0.00804
PSO-LS	0.714	0.273	0.0119	1	0	0.00568
MP-RBRS	0.667	0.227	0.0183	0.974	0.0333	0.0078
SS	0.556	0.364	0.0172	0.906	0.1	0.00459
	j30_4 (RF = 1.0)			j60_4 (RF = 1.0)		
algorithm	ER	ONVGR	GD	ER	ONVGR	GD
PLA	0.385	0.696	0.00335	0.409	0.565	0.00241
GLS	0.88	0.13	0.00774	1	0	0.016
PSO-LS	0.88	0.13	0.00677	0.951	0.0435	0.00689
MP-RBRS	1	0	0.00761	0.977	0.0217	0.00462
SS	0.875	0.087	0.0133	0.513	0.413	0.00252

Our experimental findings indicate that the number of reference solutions are significantly influenced by problem size and RF; for instance, the number of reference solutions are 4, 14, 20 and 10, 22, 39 for 30-activity and 60-activity instances with RF = 0.25, 0.50, 0.75, 1, respectively. Similar results are observed in the number of representative solutions obtained by each algorithm.

The convergence measures of the five algorithms are given in Table 3. Clearly, PLA excels in ER and ONVGR measures for all instances, and the discrepancy becomes more pronounced as RF and problem size increase. Algorithm SS ranks second in these two performance metrics. As for GD metric, PLA on the whole yields the best performance for large RF and problem sizes, followed by SS, PSO-LS and MP-RBRS, and finally GLS. The PLA provides a flexible design that integrates various types of algorithms, and utilizes their advantages. Our experimental results have shown the superiority of the PLA's hybrid structure.

4.3 Diversity Measure

Diversity is another important characteristic for measuring the quality of a nondominated solution set. One popular measure for diversity is *spread* (Deb, 2001), which calculates a relative distance between consecutive representative (nondominated) solutions. It takes care of the extent of the spread also and requires a reference set P* to compute the measure. Mathematically, it may be expressed as:

$$Spread\ (A) = h_i = Min_{j=1}^{|P^*|} \sqrt{\sum_{k=1}^{2} \left(\frac{g_k^i - g_k^j}{g_k^{\max} - g_k^{\min}} \right)^2} \tag{5}$$

where h_i is the distance between two consecutive solutions, $\bar{h}$ is the mean of all h_i's, h_m^e is the distance between the extreme solutions of P^* and A corresponding to the mth objective function. The value of this measure is zero for an ideal distribution of solution set but it can be more than 1 as well for a bad distribution of solution set. Table 4 presents the *spread* measure of the five algorithms. It shows that the diversity of the solutions obtained by the five algorithms does not substantially deviate from each other. However, GLS appears to perform better than the others for the 60-activity instances.

Table 4. Spread measure of the algorithms.

	Spread							
algorithm	j30_1	j30_2	j30_3	j30_4	j60_1	j60_2	j60_3	j60_4
PLA	0.455	0.448	0.488	0.352	0.837	0.679	0.708	0.678
GLS	0.216	0.788	0.564	0.402	0.657	0.713	0.68	0.582
PSO-LS	0.622	0.624	0.538	0.393	0.778	0.815	0.66	0.678
MP-RBRS	0.359	0.421	0.601	0.398	0.688	0.69	0.726	0.702
SS	0.423	0.334	0.475	0.554	0.766	0.841	0.766	0.622

5. CONCLUSIONS AND FUTURE RESEARCH

This paper studies the discrete makespan/resource tradeoffs RCPSP with an aim to provide the project management with a set of competitive alternatives when negotiating projects with clients. The problem solving approach comprises two phases: (1) construct a main algorithm framework, and (2) select an algorithm that will work efficiently and effectively under this framework. Five algorithms are developed to solve the RCPSP with the objective of minimizing project makespan. An experiment was conducted to test which composite algorithm will provide the most satisfactory result to the project management. The experimental results show that the new hybrid approach, population learning algorithm, performs better than the other three algorithms, particle swarm optimization, genetic local search, regret-based biased random sampling multi-pass, in terms of pair-wise comparison as well as convergence measure. However, the spread metric does not reveal any significant difference in diversity between the four algorithms.

This study is focused on developing a promising algorithm for solving the single-mode makespan/resource tradeoffs RCPSP. A more attractive research work would be to extend the current study to a multi-mode case; i.e., each activity has two or more execution options due to resource/resource and resource/duration tradeoffs. Such problems, regardless of whether the resource types are discrete, continuous, or mixed, are much more challenging and useful in practice.

ACKNOWLEDGMENT

This research was supported by the National Science Council in Taiwan under grant NSC 95-2221-E-155-045.

REFERENCES

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

Blazewicz, J., Lenstra, J. K., and Rinooy Kan, A. H. G. (1983), Scheduling subject to resource constraints: Classification and complexity, *Discrete Mathematics*, **5**, 11-24.

Boctor, F. F. (1990), Some efficient multi-heuristic procedures for resource-constrained project scheduling, *European Journal of Operational Research*, **49**, 3-13.

Deb, K. (2001), *Multi-objective optimization using evolutionary algorithm*, Wiley, Chichester, 328.

Deb, K. and Jain, S. (2002), Running performance metrics for evolutionary multiobjective optimization, *Asia-Pacific Conference Simulated Evolution and Learning* (SEAL 01), Singapore, 12-20.

Debels, D., Reyck, B. D., Leus, R., and Vanhoucke, M. (2006), A hybrid scatter search/electromagnetism meta-heuristic for project scheduling, *European Journal of Operational Research*, **169**, 638-653.

Demeulemeester, E. L. and Herroelen, W. S. (2002), *Project scheduling*: *A research handbook*, Norwell, MA: Kluwer, 8.

Eberhart, R. C., Simpson, P., and Dobbins, R. (1996), *Computational Intelligence PC Tools*, Academic Press Professional, 212-226.

Erenguc, S. S. and Icmeli-Tukel, O. (1999), Integrating quality as a measure of performance in resource-constrained project scheduling problems, edited by Jan Weglarz, *Project Scheduling*: *Recent Models*, *Algorithms and Applications*, Kluwer Academic Publishers, USA, 433-450.

Erenguc, S. S., Ahn, T., and Conway, D. G. (2001), The resource constrained project scheduling problem with multiple crashable modes: An exact solution method, *Naval Research Logistics*, **48**, 107-127.

Glover, F. (1977), Heuristics for integer programming using surrogate constraints, *Decision Sciences*, **8**, 156-166.

Glover, F. and Kochenberge, G. A. (2003), Handbook of Metaheuristics, editors, Kochenberge, Kluwer Academic Publishers.

Herroelen, W., Demeulemeester, E. and De Reyck, B. (1999), A classification scheme for project scheduling, edited by J. Weglarz, *Project Scheduling*: *Recent Models, Algorithms and Applications*, Academic Publishers, USA, 1-26.

Hsu, C. C. and Kim, D. S. (2005), A new heuristic for the multi-mode resource investment problem, *Journal of the Operational Research Society,* **56**, 406-413.

Jedrzejowicz, P. (1999), Social learning algorithm as a tool for solving some different scheduling problems, *Foundation of Computing and Decision Science*, **24**(2), 51-66.

Jedrzejowicz, P. and Ratajczak, E. (2006), Population learning algorithm for the resource-constrained project scheduling, *Perspectives in Modern Project Scheduling*, edited by J. Józefowska and J. Weglarz, Springer's International Series.

Józefowska, J., Mika, M., Rózycki, R, Waligóra, G. and Węglarz, J. (1998), Local search metaheuristics for discrete-continuous schedulingproblems, *European Journal of Operational Research*, **107,** 354-370.

Keskin, B. B. and Uster, H. (2007), A scatter search-based heuristic to locate capacitated transhipment points, *Computers and Operations Research*, **34**, 3112-3125.

Kennedy, J. and Eberhart, R. C. (1995), Particle swarm optimization, In Proceedings of the 1995 *IEEE International Conference on Neural Networks*, Piscataway, New Jersey, IEEE Service Center, 1942-1948.

Kemmoé Tchomté, S. and Gourgand, M. (2009), Particle swarm optimization: A study of particle displacement for solving continuous and combinatorial optimization problems, *International Journal of Pro-*

duction Economics, doi:10.1016/j.ijpe.2008.03.015.
Klein, R. (2000), Bidirectional planning: improving priority rule-based heuristics for scheduling resource-constrained projects, *European Journal of Operational Research*, **127**, 619-638.
Kolisch, R. and Drexl, A. (1996), Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation, *European Journal of Operational Research*, **90**, 320-333.
Kolisch, R. and Hartmann, S. (2006), Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling: An update, *European Journal of Operational Research*, **174**, 23-37.
Kolisch, R. and Sprecher, A. (1996), PSPLIB-A project scheduling problem library, *European Journal of Operational Research*, **96**, 205-216.
Li, K. Y. and Willis, R. J. (1992), An iterative scheduling technique for resource-constrained project scheduling, *European Journal of Operational Research*, **56**, 370-379.
Marti, R., Laguna, M., and Glover, F. (2006), Principle of scatter search, *European Journal of Operational Research*, **169**, 359-372**.**
Suganthan, P. N. (1999), Particle swarm optimizer with neighbourhood operator, in *Proceedings of the Congress on Evolutionary Computation*, (Washington DC, USA), IEEE Service Center, Piscataway, NJ, 1958-1961.
Tormos, P. and Lova, A. (2003), An efficient multi-pass heuristic for project scheduling with constrained resources, *International Journal of Production Research*, **41**(5), 1071-1086.
Valls, V., Ballestin, F., and Quintanilla, S. (2005), Justification and RCPSP: A technique that pays, *European Journal of Operational Research*, **165**, 375-386.
Van Veldhuizen, D. A. (1999), *Multiobjective evolutionary algorithms: classifications*, *analyses*, *and new Innovations*, PhD thesis, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wri-ght-Patterson AFB, Ohio.
Vanhoucke, M., Demeulemeester, E., and Herroelen, W. (2002), Discrete time/cost trade-offs in project scheduling with time-switch constraints, *Journal of the Operational Research Society*, **53**(7), 741-751.
Waligóra, G. (2008), Discrete-continuous project scheduling with discounted cash flow-A tabu search approach, *Computers and Operations Research*, **35**, 2141-2153.
Yamashita, D. S., Armentano, V. A., and Laguna M. (2006), Scatter search for project scheduling with resource availability cost, *European Journal of Operational Research*, **169**, 623-637.
Zitzler, E. and Thiele, L. (1999), Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, *IEEE Transactions on Evolutionary Computation*, **3**(4), 257-271.
Zhang, H., Li, X., Li, H., and Hung, F. (2005), Particle swarm optimization-based schemes for resource-constrained project scheduling, *Automation in Construction*, **14**, 393-404.