Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 519414, 9 pages
http://dx.doi.org/10.1155/2014/519414

Research Article

An Efficient Particle Swarm Optimizer with Application to Man-Day Project Scheduling Problems

Ruey-Maw Chen[1] and Frode Eika Sandnes[2]

[1] *National Chin-Yi University of Technology, No. 57, Section 2, Zhongshan Road, Taiping, Taichung 41170, Taiwan*
[2] *Oslo and Akershus University College of Applied Sciences, P.O. Box 4, St. Olavs Plass, 0130 Oslo, Norway*

Correspondence should be addressed to Ruey-Maw Chen; raymond@mail.ncut.edu.tw

Received 25 February 2014; Accepted 8 April 2014; Published 6 May 2014

Academic Editor: Her-Terng Yau

The multimode resource-constrained project scheduling problem (MRCPSP) has been confirmed to be an NP-hard problem. Particle swarm optimization (PSO) has been efficiently applied to the search for near optimal solutions to various NP-hard problems. MRCPSP involves solving two subproblems: mode assignment and activity priority determination. Hence, two PSOs are applied to each subproblem. A constriction PSO is proposed for the activity priority determination while a discrete PSO is employed for mode assignment. A least total resource usage (LTRU) heuristic and minimum slack (MSLK) heuristic ensure better initial solutions. To ensure a diverse initial collection of solutions and thereby enhancing the PSO efficiency, a best heuristic rate (HR) is suggested. Moreover, a new communication topology with random links is also introduced to prevent slow and premature convergence. To verify the performance of the approach, the MRCPSP benchmarks in PSPLIB were evaluated and the results compared to other state-of-the-art algorithms. The results demonstrate that the proposed algorithm outperforms other algorithms for the MRCPSP problems. Finally, a real-world man-day project scheduling problem (MDPSP)—a MRCPSP problem—was evaluated and the results demonstrate that MDPSP can be solved successfully.

1. Introduction

The well-known resource-constrained project scheduling problem (RCPSP) is a combinatorial optimization problem where activities are scheduled such that the *makespan* is minimized, while satisfying given precedence constraints between the activities and resources. However, a more realistic project scheduling model termed multimode resource-constrained project scheduling problem (MRCPSP) is studied herein. The term multimode indicates that the project scheduling problem has varying operation modes available for each activity; each mode includes combinations of resource requirements and processing durations. Thus, different operation mode assignments for activities would yield different project scheduling results. The MRCPSP is subjected to precedence and resources constraints. Hence, the scheduling target of MRCPSP is to find an adequate mode assignment for each activity and determine a satisfactory activity priority while satisfying the constraints, thus minimizing the makespan.

Scheduling problems such as job-shop, flow-shop, and vehicle routing have been studied intensively, and are confirmed to be NP-complete in their general forms. Both RCPSP and MRCPSP have also been proved to be NP-hard [1]. Therefore, many studies have attempted solving scheduling problems using neural networks [2], metaheuristics based algorithms including Tabu search (TS) [3, 4], simulated annealing (SA) [5], genetic algorithms (GA) [6], ant colony optimization (ACO) [7], and particle swarm optimization (PSO) [2, 8, 9].

Several metaheuristics have been proposed for solving MRCPSP. Ranjbar et al. [10] solved small scale instances of MRCPSP based on a scatter search algorithm. Combinational particle swarm optimization (CPSO) was proposed by Jarboui et al. [11]. Additionally, Zhang et al. [12] applied two conventional PSOs to construct solutions for MRCPSP. PSO is a promising and applicable methodology for a variety of combinatorial problems and diverse scheduling problems as well as other applications. Particle swarm optimization (PSO) was first proposed by Kennedy and Eberhart [13]. Many derivatives of PSO have since been examined to significantly improve functionality, of which standard PSO [14] is probably the best known. Another variation called discrete PSO (DPSO) was proposed by Kennedy and Eberhart [13]. Solving MRCPSP is regarded as solving two subproblems (mode assignment and activity priority determination); hence, dual PSOs (based on constriction) are proposed to cope with these two subproblems. A discrete PSO is adopted to decide the discrete mode.

Moreover, conventional PSOs with global communication topologies usually lead to premature convergence in local optima. Hence, a modified global best experience on the basis of local communication topology to ensure stable convergence was introduced [14], and yet the convergence is slow. Therefore, a swarm communication topology with random links (rand-link communication topology) is presented herein to increase the PSO efficiency. Restated, a trade-off mechanism between local exploitation and global exploration abilities is proposed. Additionally, two heuristics, least total resource usage (LTRU) and the minimum slack (MSLK), are used to further enhance effectiveness. The two heuristics are used to achieve better initial solutions and thus speed up the search. However, an initial set of diverse solutions would increase the chances of finding better results. Hence, a heuristic rate (HR) is explored.

To improve the effectiveness and efficiency of the proposed scheme, the largest scale scheduling case of MRCPSP in PSPLIB [15] was tested to find optimal parameters. To verify the performance of the proposed scheme, all cases of the MRCPSP benchmark instances were evaluated. Performance comparisons between algorithms were conducted. Finally, the experimental results demonstrate that the proposed scheme outperforms other schemes and is efficient in solving MRCPSP class problems. In South East Asia, project managers often use man-day as a project scheduling and management unit. Different man-day combinations are considered as a different operation modes. Hence, this man-day project scheduling problem (MDPSP) can be regarded as a MRCPSP problem. A real-world MDPSP case was finally tested; the optimal operation mode for every task is provided and the minimum completion time of the project is also given. The project manager is then able to adjust the manpower based on the results. The remainder of the paper is organized as follows. Section 2 provides descriptions on MRCPSP and MDPSP. Section 3 introduces particle swarm optimization, discrete particle swarm optimization, and standard particle swarm optimization. The proposed dual PSO approach with heuristics and rand-link communication topology to solve scheduling problems is also presented in Section 3. In Section 4, experimental results and comparisons are demonstrated. Finally, the study is summarized in Section 5.

2. Scheduling Problems

2.1. MRCPSP. A MRCPSP instance includes precedence and resource constraints based on an operation mode where different combinations produce different schedules. The multimode scheduling problem is defined as follows.

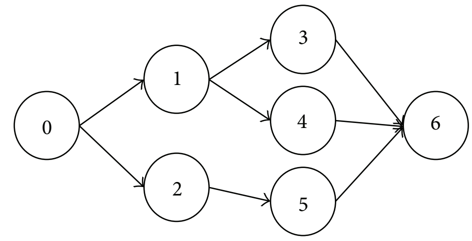


Figure 1: Activity on node (AON).

The MRCPSP includes n activities to be scheduled and two dummy activities represent the start and the end of the project. Let J be a set consisting of all activities, denoted $J = \{0, \ldots, n+1\}$. Accordingly, the project can be characterized by an activity-on-node (AON). MRCPSP involves precedence and resource constraints; some tasks in MRCPSP are partially ordered. For tasks without precedence constraint, different execution sequences would result in different schedules and hence the different completion times. Meanwhile, each activity j has M_j available operation modes. The set $\text{Mode}_j = \{1, \ldots, M_j\}$ includes the available modes of activity j, and the operation mode of activity j is denoted $m_j, m_j \in \text{Mode}_j$. Moreover, each mode involves a required processing time (p_{j,m_j}) and different resource types needed for completing the activity. Hence, a task with different operation modes would yield a different schedule. When the jth activity performs n mode m_j, the start time of activity is s_j and processing time is $p_{j,m}$. Hence the finish time of the activity is denoted by $f_j = s_j + p_{j,m_j}$.

MRCPSP provides two types of resources: renewable resources and nonrenewable resources. For each renewable resource, a fixed amount of resources are provided during each time period. The available amount of each nonrenewable resource is constant throughout the entire project. The amount of renewable resources K required by activity j at mode m_j is denoted by $r^K_{j,m}$ and the required nonrenewable resource q by activity j at mode m_j is represented by $n^q_{j,m}$. However, all consumed renewable resources at any time period should not exceed what is provided by the system; that is, $\sum_{j \in A(t)} r^k_{j,m_j} \le R_k, R_K$ is the available amount of renewable resource type K, and $A(t)$ denotes the set in which activities are being processed at time t; all used nonrenewable resources in the whole project should not be more than system-supplied; that is, $\sum_{j \in J} n^q_{j,m_j} \le N_q, N_q$ is the available amount of the qth nonrenewable resource. A simple example of MRCPSP in PSPLIB is given in Table 1 and Figure 1.

The schedule is said to be *feasible* when situations of both resources and precedence constraints are met; otherwise, the schedule is *infeasible*. For example, a project schedule with the activity priority list $\{1, 5, 2, 3, 4\}$ is *infeasible* since it violates a precedence constraint. Meanwhile, a project schedule with tasks' operation mode list $\{1, 1, 1, 1, 1\}$ is also *infeasible* because the overall amount of consumed nonrenewable resources exceeds the amount provided by

Table 1: Five activity instances of MRCPSP in PSPLIB.

(a)

Activity	#Modes	#Successors	Successors	
0	0	2	1	2
1	1	2	3	4
2	1	1	5	
3	1	1	6	
4	1	1	6	
5	3	1	6	
6	1	0		

(b)

Activity	Mode	Duration	Renewable	Nonrenewable
0	0	0	0	0
1	1	1	2	4
2	1	3	2	5
3	1	4	1	1
4	1	3	3	5
	1	1	3	6
5	2	2	2	5
	3	3	1	4
6	0	0	0	0
Available resources			**4**	**20**

the system. Moreover, the project schedules with the same activity priority list, say {1, 2, 3, 4, 5}, but different operation mode lists {1, 1, 1, 1, 3} and {1, 1, 1, 1, 2} would yield different *makespans,* namely, 8 and 7, respectively. The project schedules with the same operation mode list, say {1, 1, 1, 1, 2}, but different activity priority lists {1, 3, 4, 2, 5} and {1, 2, 3, 4, 5} would also result in different *makespans, that is,* 9 and 7, respectively. Therefore, different combinations of mode list and priority list would result in different project schedules. To obtain a near optimal solution, the project requires a satisfactory activity priority list and an adequate mode list. In other words, the target of this investigation for solving multimode project scheduling is to find the optimal combination of these two lists so as to minimize the *makespan*. Overall, favourable mode and activity lists yield good solutions.

2.2. MDPSP. The man-day project scheduling problem (MDPSP) is common in South East Asia. In MDPSP, one man-day is utilized as a management unit in estimating and planning a project, for example, a project manager may estimate that a project task requires four man-days. Thus, the task may need four people to work for one day, two people to work for two days, or one person to work for four days. Every man-day combination is considered as an operation mode; every task has different operation modes. Therefore, this man-day project scheduling problem (MDPSP) can be regarded as a MRCPSP problem. Similarly, the MDPSP also involves precedence constraints; that is, a task can be executed only if all its predecessor tasks have been completed, and every task is nonpreemptive. Moreover, the people in the work team are considered renewable resources.

Task items:

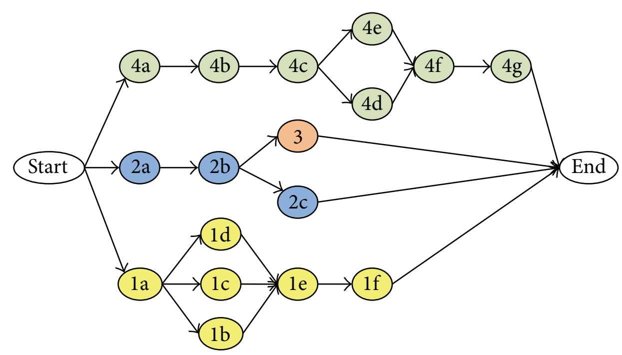
(1) Power configuration
- (a) Power panel "C-P" × 2 sets mounting (2 man-day)
- (b) Power panel input total power configuration (6 man-day)
- (c) UPS input and output power configuration (2 man-day)
- (d) UPS output circuit 110 V power configuration-24 circuits(8 man-day)
- (e) Rack and UPS move in (4 man-day)
- (f) Rack mounting and UPS installation (2 man-day)

(2) Fiber optic backbone
- (a) Pipeline dredge and link (4 man-day)
- (b) Single mode fiber optic 8 C × 2 laying (16 man-day)
- (c) Fiber optic finishing and marking (2 man-day)

(3) Fiber fusion and testing (2 man-day)

(4) UTP cat-6 network
- (a) Layout confirmation (4 man-day)
- (b) Pipeline dredge confirmation(16 man-day)
- (c) Laying (96 man-day)
- (d) DVO fabrication and marking (64 man-day)
- (e) Patch panel wiring(16 man-day)
- (f) Patch cord labeling (8 man-day)
- (g) Finishing rack and testing (12 man-day)

(a) Task items in the real-world project



(b) Corresponding AON

Figure 2: A real-world MDPSP example.

The nonrenewable resource is supposed to be enough. A real-world MDPSP example is shown in Figure 2—a real network construction project. Figure 2(a) lists the required task items and man-days needed for every task; Figure 2(b) displays the corresponding AON.

3. Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) was first proposed by Kennedy and Eberhart in 1995 [13]. PSO is a multiagent general metaheuristic and has been widely applied to many complex and NP-hard problems. PSO is initialized with a population of randomly positioned particles and searches for the position with the best fitness. The particle position is the representative of a solution or schedule correlated with fitness (makespan) in this investigation.

In each generation or iteration, every particle moves to a new position guided by velocity and then the fitness of the particles is calculated. There are two experience positions used in PSO for updating the velocity: one is the global experience position of all particles, which remembers the global best solution obtained by all particles; the other is each particle's individual experience, which remembers the best position that particle has been moved to. Formally, let an N-dimensional space have M particles. For the ith particle ($i = 1, \ldots, M$), its position consists of N components $\mathbf{X}_i = \{X_{i1}, \ldots, X_{iN}\}$, where X_{ij} is the jth component of the position. The velocity of particle i is $\mathbf{V}_i = \{V_{i1}, \ldots, V_{iN}\}$ and the particle's individual experience is $\mathbf{L}_i = \{L_{i1}, \ldots, L_{iN}\}$. Additionally, $\mathbf{G} = \{G_1, \ldots, G_N\}$ represents the global best experience shared among all the particles. The velocity and position update rules used in the original PSO are displayed as follows:

$$V_{ij}^{\text{new}} = w \times V_{ij} + c_1 \times r_1 \times \left(L_{ij} - X_{ij}\right) + c_2 \times r_2 \times \left(G_j - X_{ij}\right),$$
$$X_{ij}^{\text{new}} = X_{ij} + V_{ij}^{\text{new}}, \tag{1}$$

where w is an inertia weight used to determine the influence of the previous velocity on the new velocity. The c_1 and c_2 are learning factors used to guide how close to the individual or global experience position, respectively. Moreover, the r_1 and r_2 are random numbers uniformly distributed in the interval $[0, 1]$, influencing the tradeoff between the global (swarm's best experience) and local (particle's best experience) exploration abilities during search. The velocity update rule and the position update rule are regarded as crossover and mutation operations of evolutionary computation, respectively. The velocity updating plays an important part in PSO while searching for a solution with better fitness. Therefore, several derivatives of PSO have been proposed to update the velocity vector. One of them is discrete version of PSO developed by Kennedy and Eberhart [22]. Another is the standard PSO proposed by Bratton and Kennedy [14]. In this work, the velocity update rule of standard PSO is the basis of our proposed dual PSO mechanism. The discrete PSO is further introduced for solving mode assignment.

3.1. Solution Encoding Scheme. The aim of the mode assignment subproblem is to generate a mode list $\mathbf{X}^M = \{X_1^M, \ldots, X_j^M\}$ (J activities), which determines the operation mode for each activity, that is, determining each activity's duration and resources. The X_j^M is the mode for activity j which has bit_j binary values. The aim of the activity priority determination subproblem is to generate the activity list $\Pr = \{0, \ldots, n+1\}$ which is obtained from activity list $\mathbf{X}^J$. Hence, the solution vector $\mathbf{X}$ is composed of two position vectors corresponding to the mode list $\mathbf{X}^M$ and activity list $\mathbf{X}^J$; that is, $\mathbf{X} = \mathbf{X}^M \cup \mathbf{X}^J$.

3.2. Standard Particle Swarm Optimization. In PSO, the balance between the global exploitation and local exploration processes is mainly controlled by the inertia weights. A suitable selection of the inertia weight w can provide a balance between the global and local exploration processes and thus requires less iteration on average to find the optimum. In this investigation, the named standard PSO (or constriction version PSO) is applied to mode assignment and activity priority determination. The standard PSO involves a constriction factor to replace inertia weight in the velocity update rule as follows:

$$V_{ij}^{\text{new}} = \chi \times \left(V_{ij} + c_1 \times r_1 \times \left(L_{ij} - X_{ij}\right) + c_2 \times r_2 \times \left(G_j - X_{ij}\right)\right), \tag{2}$$

where χ is the constriction factor used for controlling the movement velocity. This velocity update rule is suggested for its stability as indicated in [14].

3.3. Discrete PSO Encoding Scheme for Mode Assignment. For the discrete PSO, the velocity update rule of the standard PSO is as in (2). The X_{ij} is the jth component of $\mathbf{X}^M$ position vector. The value of X_{ij} is either 0 or 1. The value of individual experience PSO (L_{ij}, the jth component of particle i) or global experience PSO (G_n, the nth component) is also either 0 or 1; that is, $X_{i,n}, L_{i,n}, G_{i,n} \in \{0, 1\}$. However, the values of the velocity components are still real numbers since r_1 and r_2 are random numbers. In this work, they are limited to the interval $[-V_{\max}, V_{\max}]$. Each particle moves to a new position according to its new velocity. However, the new position generation of the discrete PSO is not the same as in the original PSO. Kennedy and Eberhart [22] advocated that the higher the velocity, the more likely to choose 1 for the corresponding position component, and low velocity favors a position value of 0. Hence, a sigmoid function is used as the probability function as shown in (3). $S(V_{ij}^{\text{new}})$ is defined as representing the probability of X_{ij}^{new} to be set to 0 or 1. To avoid the value of $S(V_{ij}^{\text{new}})$ approaching 0 or 1, a constant $V_{\max}$ is used to limit the range of V_{ij}^{new}. The resulting position value X_{ij}^{new}, either 1 or 0, is determined by $S(V_{ij}^{\text{new}})$ and a uniform distribution random variable rand in $[0, 1]$ as defined in (4). The component number of position vector for activity j (X_j^M) is determined based on M_j. If the corresponding position bits for activity j is $X_j^M = \{0\ 1\ 0\}$, then $m_j = 2$; that is, activity j is assigned to execution in operation mode 2 as follows:

$$S\left(V_{ij}^{\text{new}}\right) = \frac{1}{1 + \exp\left(-V_{ij}^{\text{new}}\right)}, \tag{3}$$

$$X_{ij}^{\text{new}} = \begin{cases} 1, & \text{rand} < S\left(V_{ij}^{\text{new}}\right) \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

3.4. Determining Activity Priority. Activity processing order greatly affects the *makespan* while meeting the given precedence constraints. Therefore, determining satisfactory activity priority is important for solving MRCPSP. Suppose that the position vector $\mathbf{X}^J$ corresponds to the activity list Pr with J components, and then each component represents the activities' precedence relations. The activity list is a priority list, that is, a permutation without repeating values. Therefore, the encoding mechanism for mode list $\mathbf{X}^M$ is not applicable. Instead, a random key scheme is applied to determine the activity priority; each component of $\mathbf{X}^J$ is associated with an integer key. The component values of $\mathbf{X}^J$ are sorted in ascending order. Next, the key is used as the activity priority, then the activity list Pr is produced. An example of a random key scheme is displayed in the following equation:

$$\begin{gathered}\text{Key} \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \\ \mathbf{X}^J = \{0.3 \;\; 3.2 \;\; 0.9 \;\; 2.6 \;\; 1.4\} \\ \text{Sorting in ascent order} \downarrow \\ \mathbf{X}^{J'} = \{0.3 \;\; 0.9 \;\; 1.4 \;\; 2.6 \;\; 3.2\} \\ \downarrow \\ \text{Priority list Pr} = \{1 \quad 3 \quad 5 \quad 4 \quad 2\}.\end{gathered} \tag{5}$$

Notably, the MRCPSP has precedence constraints; hence, the produced activity priority list Pr may result in an infeasible solution. The repair mechanism can be applied to correct activity priority list Pr to Pr′ (repaired priority list).

3.5. New Swarm Communication Topology. There have been two commonly used swarm communication topologies for the standard PSO, namely, the g_{Best} (global best) topology and the l_{Best} (local best) topology. The g_{Best} topology is the most widely examined and is based on the global topology of the network, in which every particle is able to share information with each other quickly and performs better because of its global communication ability. However, for more complex problem, the global communication ability of g_{Best} usually leads to premature convergence and becomes trapped in local optima. Hence, the l_{Best} topology was addressed and has recently attracted the researchers' attention. The structure is based on the local topology of the network such as ring or star. The feature of l_{Best} is its limited communication; every particle just communicates with a portion of the swarm. Obviously, l_{Best} has a slower rate of convergence than g_{Best}.

A new swarm communication rand-link topology is proposed. The rand-link topology is applied to avoid both slow convergence and premature convergence. It is based on the ring topology with some additional random links. The rand-link topology combines l_{Best} topology (determined by X_{i+1}, X_i, X_{i-1}) and R random selected particles.

3.6. Heuristics Rate for Initial Solutions. By starting the search from a better position, the probability of finding a near optimal solution and more rapid convergence is increased. A heuristic is an experience-based strategy or technique, the aim of which is usually to give better quality solutions over time. In PSO, randomly assigned initial particle position may reduce the efficiency of the algorithm. Hence, to effectively solve the problem, most studies would consider how to generate a good initialization population of the particles. In this study, two heuristics based on priority rules [23] are applied for generating two initial position vectors $\mathbf{X}^M$ and $\mathbf{X}^J$, namely, the least total resource usage (LTRU) heuristic and the minimum slack (MSLK) heuristic. The LTRU heuristic is based on the mode priority rule, which determines each mode i of jth activity with a priority value $1/v_j(i)$; the definition of $v_j(i)$ is listed in (6) as follows:

$$v_j(i) = \sum_{k=1}^{K} \left(r_{j,i}^k \times p_{j,i} \right). \tag{6}$$

According to (6), an operation mode requesting more renewable resources (k) and requiring longer processing time would have a larger $v_j(i)$ value and thus have a lower mode priority.

In the activity priority determination problem, the activity priority rule assigns a priority value for each activity j; the MSLK is used in this work. The activity priority value is determined by $1/p(j)$, the definition of $p(j)$ is displayed below in (7) as follows:

$$p(j) = \left(\text{LS}_j - \text{ES}_j \right), \tag{7}$$

where LS_j and ES_j are the latest start time and the earliest start time of activity j, respectively; these are obtained by the critical path method in this study. Therefore, the latest start time close to the earliest start time has the higher activity priority. However, applying this heuristics in full to the initial solutions would lead to premature convergence. To avoid this problem, diverse initial solutions are ensured by proposing a heuristic rate (HR) to decide the ratio of the initial solutions generated by heuristics. The design is as follows:

$$\begin{aligned}&\text{initial solutions}\\ &\quad = \begin{cases} \text{particles} \times \text{HR}, & \text{heuristics generated} \\ \text{particles} \times (1 - \text{HR}), & \text{random generated.} \end{cases}\end{aligned} \tag{8}$$

3.7. Fitness Design. The scheduling target of MRCPSP is to find an adequate schedule while following constraints and thus minimizing the *makespan,* that is, minimizing the completion time of the dummy activity ($n + 1$). The fitness of a solution is defined as the reciprocal of the *makespan*, as indicated in (9) as follows:

$$\text{fitness} = (\text{makespan})^{-1}. \tag{9}$$

An obtained schedule violate constraint is an infeasible solution. Therefore, a penalty mechanism is designed for infeasible solution. Consider that the resource requirement (n^p) of each activity (j) is based on the predefined operation modes ($m_j \in \text{Mode}_j$) and that the amount of the given

Initialize particles' positions ($\mathbf{X} = \mathbf{X}^M \cup \mathbf{X}^J$) by applying heuristics: LTRU (for $\mathbf{X}^M$) and MSLK (for $\mathbf{X}^J$).
Iteration Loop
  For each particle i in the swarm do:
  Update the velocity vector $\mathbf{V}_i^M$ and position vector $\mathbf{X}_i^M$
    according to (2) and (4), respectively.
  Update the velocity vector $\mathbf{V}_i^J$ and position vector $\mathbf{X}_i^J$
    according to (2)
  Calculate the mode list vector M based on $\mathbf{X}_i^M$ (4).
  Calculate the activity list vector Pr by applying the random
    key scheme to and constructing a new feasible
    precedence for the process order Pr' by repair mechanism.
  Calculate the particle's fitness (based on vector M and vector Pr').
  Update L_i.
  Update G (g_{Best}, l_{Best} or rand-link).
Until the End condition is reached, return solution.

Algorithm 1: The proposed dual PSO.

available resources is N_p. Therefore, the penalty mechanism is designed to assign a nonzero penalty value on infeasible solutions. The penalty value is calculated based on the degree to which the resource constraints are exceeded; hence, the penalty value (PV) calculation is designed as follows:

$$\mathrm{PV} = \sum_{p \in N} \max \left\{ 0, \sum_{j \in J} n^p_{j, m_j \in \mathrm{Mode}_j} - N_p \right\}. \qquad (10)$$

The fitness function involving the penalty value, PV, is then redefined as follows:

$$\text{fitness} = \frac{1}{(\text{makespan} + \mathrm{PV})}. \qquad (11)$$

The proposed particle swarm optimization is summarized in Algorithm 1.

4. Experimental Results and Comparisons

To evaluate the performance of the proposed scheme, a test on a benchmark was conducted prior to solving real-word man-day project scheduling problems. Test instances in the well-known project scheduling problem library (PSPLIB) [15] were simulated. Simulation instances in the interested PSPLIB include scheduling problems with 10, 12, 14, 16, 18, 20, and 30 nondummy activities cases (denoted by J10 through J30); each case has 640 instances. However, some instances have no feasible solutions. Therefore, every case has a different number of feasible instances (ex. 536, 547, 551, 550, 552, 554, and 552 instances for the J10, J12, J14, J16, J18, J20, and 552 instances, resp.). To compare algorithm performance, the algorithm was tested with 5000 evaluations as the stop condition. Meanwhile, the solution quality is measured by evaluating the ratio of optimal solutions (OPT) found which is calculated using (12); "best_i" represents the best solution found for instance i. If the close to optimal *makespans* for instances are known, then the "best" is the close to optimal solution (J10 to J20) provided in PSPLIB. However, close to optimal *makespans* for some instances are unknown, and then lower bounds (the best known solution found for J30) are used instead as the "best" solution as follows:

Table 2: Comparison of different random links.

Random links	Total	OPT
2	283/552	51.27%
3	291/552	52.72%
4	**304/552**	**55.07%**
5	290/552	52.54%
6	298/552	53.97%

$$\mathrm{OPT} = \left(\frac{\sum_{i \in \text{instances}} \mathrm{best}_i}{|\text{instances}|} \right) \times 100\%. \qquad (12)$$

To determine the close to optimal heuristics rate (HR) and random links, tests were performed on the largest case (J30). Too many or too few random links would approach global or local communication topologies; hence, the best random links has to be determined. To obtain diverse initial solutions, the best HR was obtained through tests. The simulation results demonstrated that 4 random links yield good results while HR = 20% gives the best results, as illustrated in Tables 2 and 3. Initial solutions generated without heuristics yield fewer close to optimal solutions. A high HR may therefore lead to a local optimum since the algorithm converges prematurely.

Accordingly, the parameter settings applied for performance comparison are $c_1 = c_2 = 2.0$, $\chi = 0.73$, HR = 20%, and Random-links = 4 for 20 particles with 5000 schedules. Table 4 shows the simulation results of all 552 instances of the J30 case. In Table 4, "Dev. BKS (%)" shows the average deviation from the best known solutions (BKS). However, as comparison of "Dev. BKS" is not always possible, the percentage increase of the project duration above the critical path (CP) is also indicated, "Incr. CP (%)." In the last two columns ("Equal (%)" and "Worse (%)") of the table, the percentages of instances which result in equal and worse

Table 3: Comparison of 0%–100% HR.

HR	Total	OPT
0%	304/552	55.07%
10%	322/552	58.33%
20%	**337/552**	**61.05%**
30%	324/552	58.70%
40%	326/552	59.06%
50%	323/552	58.51%
60%	320/552	57.97%
70%	313/552	56.70%
80%	319/552	57.79%
90%	317/552	57.43%
100%	322/552	58.33%

Table 4: Comparison of algorithms on J30.

%	Dev. BKS	Incr. CP	Worse	Equal
This work*	**0.96**	**13.45**	**26.45**	**73.30**
Peteghem and Vanhoucke [16]	1.08	13.75	29.00	71.00
Chiang et al. [17]	2.60	N/A	27.36	72.64
Józefowska et al. [18]	11.76	N/A	74.40	25.60

results than the best known solution are shown. The "Dev. BKS" and "Incr. CP" are defined as listed in (13) and (14), respectively. Consider the following:

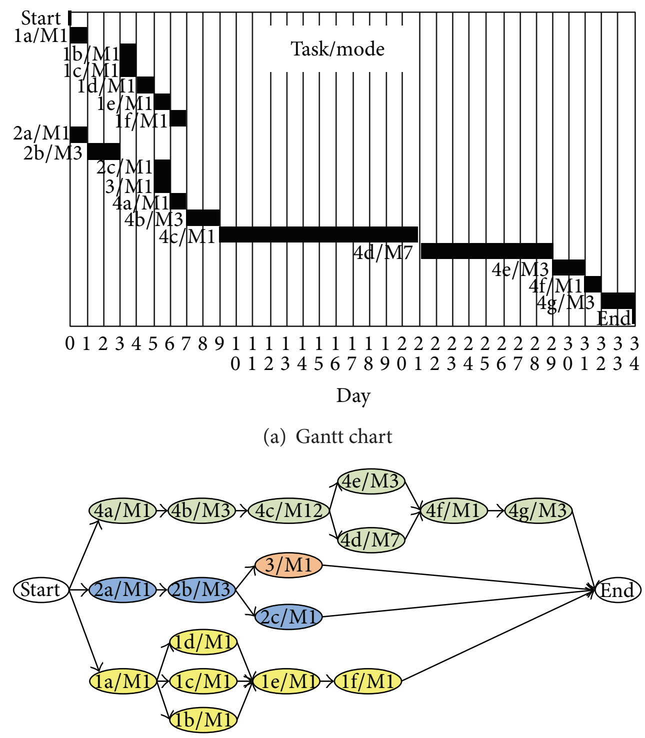
$$\text{Dev. BKS} = \frac{\sum_{i\in\text{instances}} \left(\left(\left(\text{fitness}_i - \text{best}_i\right)/\text{best}_i\right) \times 100\%\right)}{|\text{instances}|}, \quad (13)$$

$$\text{Incr. CP} = \frac{\sum_{i\in\text{instances}} \left(\left(\left(\text{fitness}_i - \text{CP}_i\right)/\text{CP}_i\right) \times 100\%\right)}{|\text{instances}|}. \quad (14)$$

Meanwhile, a comparison of the different algorithms on the J10 to J20 datasets of PSPLIB was made. Table 5 displays the simulation results of all instances of the J10 to J20 cases, with average deviations provided. In the second part of the table, the percentage of optimal solutions found for each case is also presented.

Table 4 shows that the proposed scheme yields mean deviation of 0.96% from the best known solutions, a 13.45% average increase of the project duration above the minimal critical path, and 73.3% of the best known solutions found for solving the largest case, J30. Meanwhile, the proposed scheme gives mean deviations of 0.01%, 0.07%, 0.15%, 0.15%, 0.34%, and 0.35% from the optimal solutions for J10, J12, J14, J16, J18, and J20, respectively, as listed in Table 4.

According to the simulation results, the proposed particle swarm optimizer efficiently finds near optimal solutions to the MRCPSP problem; hence, a real-world MDPSP example (Figure 2) was further investigated. In this real-world example, every task has different operation modes. For example, task 4f claims 8-man-day operation, there are 4 modes for task 4f, that is, 8/1, 4/2, 2/4 and 1/8 (man/day), corresponding to modes 1, 2, 3, and 4, respectively. The available renewable



(a) Gantt chart

(b) Task/mode assignment

Figure 3: Resulting Gantt chart and operation mode of the eight-people work team project.

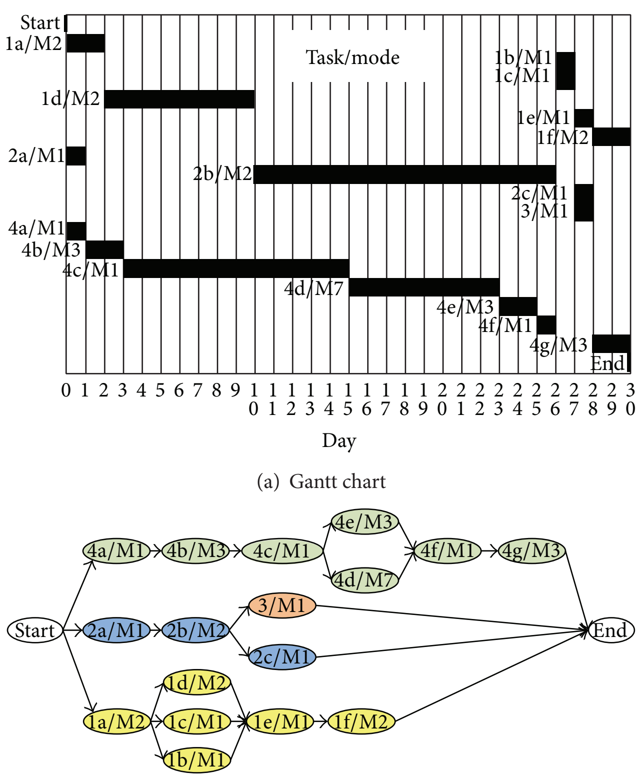
resource (manpower) in the project includes an eight-people work team. Task 2b claims 16-man-day operation, there are 1/16, 2/8, 4/4, 8/2, and 16/1 (man-day) modes. However, 16/1 operation mode requires 16 people to finish the task; it violates the available manpower resource. Therefore, 16/1 operation mode is excluded in the implementation. The simulation results are shown in a Gantt chart (see Figure 3(a)) and the best operation mode (see Figure 3(b)). In Figure 3, every task is associated with an operation mode and displayed by task/mode; for example, 4d/M7 represents task 4d executes in mode 7. These simulation results indicate that to finish the network construction project with an eight-people work team, the minimum completion time is 34 days. The project manager can adjust the required manpower based on the resulting schedule. Figures 4(a) and 5(a) display the resulting Gant charts based on the nine-people and ten-people work team alternatives. The corresponding project schedule can be shrunk to 30 or 27 days, respectively, when one or two more workers are involved into the work team. Restated, the project can be completed ahead when workers increase. Moreover, the corresponding operation mode for each task would be different accordingly as displayed in Figures 4(b) and 5(b).

5. Conclusions

MRCPSP has been confirmed to be an NP-hard optimization problem. The MRCPSP is treated as a two-part problem comprising the mode assignment subproblem and the activity

Table 5: Comparison of algorithms on J10 to J20.

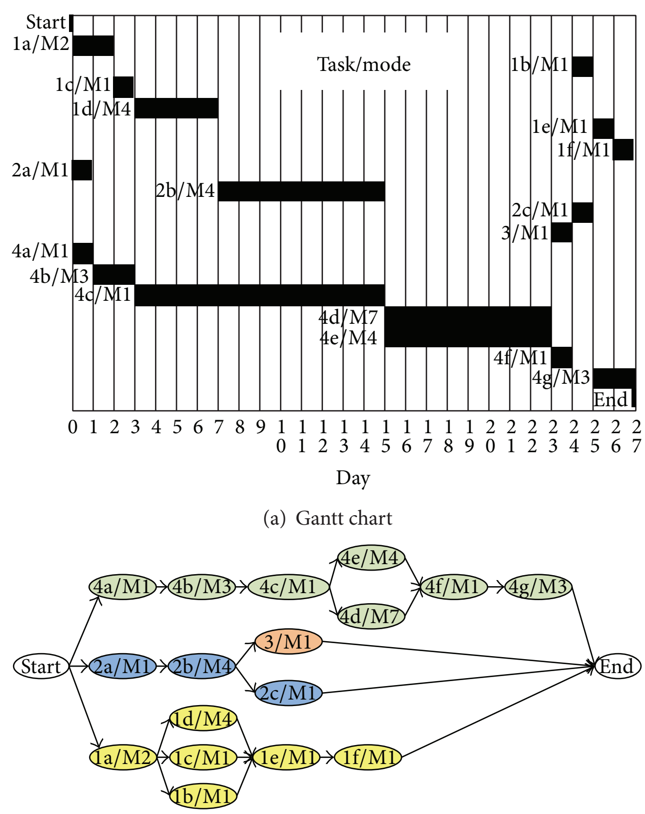
Dev. BKS (%)	Instance set					
	J10	J12	J14	J16	J18	J20
This work*	**0.01**	**0.07**	**0.15**	**0.15**	**0.34**	**0.35**
Wauters et al. [19]	0.05	0.08	0.23	0.30	0.53	0.70
Peteghem and Vanhoucke [16]	**0.01**	0.09	0.22	0.32	0.42	0.57
Lova et al. [20]	0.06	0.17	0.32	0.44	0.63	0.87
Ranjbar et al. [10]	0.18	0.65	0.89	0.95	1.21	1.64
Jarboui et al. [11]	0.03	0.09	0.36	0.44	0.89	1.10
Chiang et al. [17]	0.34	N/A	N/A	N/A	N/A	1.79
Alcaraz et al. [21]	0.24	0.73	1.00	1.12	1.43	1.91
Józefowska et al. [18]	1.16	1.73	2.60	4.07	5.52	6.74
Optimal (%)						
This work*	**99.81**	**98.35**	**96.19**	**95.64**	**90.76**	**90.25**
Peteghem and Vanhoucke [16]	99.63	98.17	94.56	92.00	88.95	85.74
Chiang et al. [17]	**99.81**	N/A	N/A	N/A	N/A	88.27
Józefowska et al. [18]	85.60	80.30	66.40	54.70	43.05	35.7



(a) Gantt chart

(b) Task/mode assignment

Figure 4: Resulting Gantt chart and operation mode of the nine-people work team project.



(a) Gantt chart

(b) Task/mode assignment

Figure 5: Resulting Gantt chart and operation mode of the ten-people work team project.

determination subproblem. Therefore, this study proposes a dual PSO scheme based on the standard PSO to efficiently solve the MRCPSP. In the proposed scheme, a random link topology was suggested to help avoid slow and premature convergence, and 4 random links yield the best results. Meanwhile, the suggested dual PSO scheme also involves the LTRU and MSLK heuristics to generate good initial particle positions. Moreover, the best heuristics rate (HR) 20% is verified. The performance comparisons between different algorithms are demonstrated in Tables 4 and 5. Table 4 shows that the proposed scheme yields minimum Dev. BKS (0.96%)

and minimum Incr. CP (13.45%), and maximum percentage of the best known solutions can be found (73.30%) for the J30 case. Meanwhile, minimum Dev. BKS for J10 to J20 (0.01%, 0.07%, 0.15%, 0.15%, 0.34%, and 0.35%) is provided as listed in Table 5. Accordingly, the experimental results demonstrated that the proposed scheme for solving MRCPSP outperforms other schemes in the literature. A real-world MDPSP was successfully solved; the optimal operation mode for each task is provided and the minimum completion time of the project can be obtained as indicated in Figures 3, 4, and 5. These resulting outcomes offer important information to project managers to make adjustment on the project.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was partly supported by the National Science Council, Taiwan, under Contract NSC 102-2221-E-167-018.

References

[1] J. Blazewicz, J. K. Lenstra, and A. H. G. R. Kan, "Scheduling subject to resource constraints: classification and complexity," *Discrete Applied Mathematics*, vol. 5, no. 1, pp. 11–24, 1983.

[2] F. Zhao, Y. Hong, D. Yu, Y. Yang, and Q. Zhang, "A hybrid particle swarm optimisation algorithm and fuzzy logic for process planning and production scheduling integration in holonic manufacturing systems," *International Journal of Computer Integrated Manufacturing*, vol. 23, no. 1, pp. 20–39, 2010.

[3] F. Glover, "Tabu Search—part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[4] F. Glover, "Tabu Search—part I," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.

[5] V. F. Yu, S.-W. Lin, and S.-Y. Chou, "The museum visitor routing problem," *Applied Mathematics and Computation*, vol. 216, no. 3, pp. 719–729, 2010.

[6] O. Sinnen, L. A. Sousa, and F. E. Sandnes, "Toward a realistic task scheduling model," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 3, pp. 263–275, 2006.

[7] S.-T. Lo, R.-M. Chen, Y.-M. Huang, and C.-L. Wu, "Multiprocessor system scheduling with precedence and resource constraints using an enhanced ant colony system," *Expert Systems with Applications*, vol. 34, no. 3, pp. 2071–2081, 2008.

[8] R.-M. Chen and C.-M. Wang, "Project scheduling heuristics-based standard PSO for task-resource assignment in heterogeneous grid," *Abstract and Applied Analysis*, vol. 2011, Article ID 589862, 20 pages, 2011.

[9] R.-M. Chen, C.-L. Wu, C.-M. Wang, and S.-T. Lo, "Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB," *Expert Systems with Applications*, vol. 37, no. 3, pp. 1899–1910, 2010.

[10] M. Ranjbar, B. de Reyck, and F. Kianfar, "A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling," *European Journal of Operational Research*, vol. 193, no. 1, pp. 35–48, 2009.

[11] B. Jarboui, N. Damak, P. Siarry, and A. Rebai, "A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems," *Applied Mathematics and Computation*, vol. 195, no. 1, pp. 299–308, 2008.

[12] H. Zhang, C. M. Tam, and H. Li, "Multimode project scheduling based on particle swarm optimization," *Computer-Aided Civil and Infrastructure Engineering*, vol. 21, no. 2, pp. 93–103, 2006.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, December 1995.

[14] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 120–127, April 2007.

[15] R. Kolisch and A. Sprecher, "PSPLIB—a project scheduling problem library," *European Journal of Operational Research*, vol. 96, no. 1, pp. 205–216, 1997.

[16] V. V. Peteghem and M. Vanhoucke, "A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 201, no. 2, pp. 409–418, 2010.

[17] C.-W. Chiang, Y.-Q. Huang, and W.-Y. Wang, "Ant colony optimization with parameter adaptation for multi-mode resource-constrained project scheduling," *Journal of Intelligent and Fuzzy Systems*, vol. 19, no. 4-5, pp. 345–358, 2008.

[18] J. Józefowska, M. Mika, R. Różycki, G. Waligóra, and J. Węglarz, "Simulated annealing for multi-mode resource constrained project scheduling," *Annals of Operations Research*, vol. 102, no. 1–4, pp. 137–155, 2001.

[19] T. Wauters, K. Verbeeck, G. V. Berghe, and P. de Causmaecker, "Learning agents for the multi-mode project scheduling problem," *Journal of the Operational Research Society*, vol. 62, no. 2, pp. 281–290, 2011.

[20] A. Lova, P. Tormos, M. Cervantes, and F. Barber, "An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes," *International Journal of Production Economics*, vol. 117, no. 2, pp. 302–316, 2009.

[21] J. Alcaraz, C. Maroto, and R. Ruiz, "Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms," *Journal of the Operational Research Society*, vol. 54, no. 6, pp. 614–626, 2003.

[22] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, pp. 4104–4108, Piscataway, NJ, USA, October 1997.

[23] J. Buddhakulsomsiri and D. S. Kim, "Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting," *European Journal of Operational Research*, vol. 178, no. 2, pp. 374–390, 2007.