Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 365697, 18 pages
doi:10.1155/2012/365697

Research Article

A Modified PSO Algorithm for Minimizing the Total Costs of Resources in MRCPSP

Mohammad Khalilzadeh,[1] Fereydoon Kianfar,[1] Ali Shirzadeh Chaleshtari,[1] Shahram Shadrokh,[1] and Mohammad Ranjbar[2]

[1] *Department of Industrial Engineering, Sharif University of Technology, Tehran 11365-8639, Iran*
[2] *Department of Industrial Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad 9177948974, Iran*

Correspondence should be addressed to Mohammad Khalilzadeh, mo.kzadeh@gmail.com

Received 6 September 2011; Revised 4 December 2011; Accepted 30 December 2011

Academic Editor: Yi-Chung Hu

We introduce a multimode resource-constrained project scheduling problem with finish-to-start precedence relations among project activities, considering renewable and nonrenewable resource costs. We assume that renewable resources are rented and are not available in all periods of time of the project. In other words, there is a mandated ready date as well as a due date for each renewable resource type so that no resource is used before its ready date. However, the resources are permitted to be used after their due dates by paying penalty costs. The objective is to minimize the total costs of both renewable and nonrenewable resource usage. This problem is called multimode resource-constrained project scheduling problem with minimization of total weighted resource tardiness penalty cost (MRCPSP-TWRTPC), where, for each activity, both renewable and nonrenewable resource requirements depend on activity mode. For this problem, we present a metaheuristic algorithm based on a modified Particle Swarm Optimization (PSO) approach introduced by Tchomté and Gourgand which uses a modified rule for the displacement of particles. We present a prioritization rule for activities and several improvement and local search methods. Experimental results reveal the effectiveness and efficiency of the proposed algorithm for the problem in question.

1. Introduction

The resource-constrained project scheduling problem (RCPSP) is the scheduling of project activities subject to precedence relations as well as renewable resource constraints with the objective of minimizing the makespan of the project. Each nonpreemptive activity in RCPSP can be done in a single mode. For more information on RCPSP and solution methods, we refer to Demeulemeester and Herroelen [1]. In the multimode RCPSP (MRCPSP), a set of allowable modes can be defined for each activity which is characterized by a constant duration and associated resource requirements. In this paper we consider MRCPSP with the objective of minimizing total costs of all resources. Two types of resources, renewable and nonrenewable, are considered. Nonrenewable resource cost of an activity is a function of its resource requirements, determined by its modes. The limited renewable resources are rented and each renewable resource is available in a predetermined sequential time period specified by its ready time and due date and is not available before the ready time. However, each renewable resource can be used after its due date with tardiness penalty cost. As the cost of renting for each renewable resource is fixed, there is no need to incorporate it into the objective function and only tardiness penalty cost is considered for each renewable resource. The MRCPSP under minimization of total costs of resources (RCPSP-TWRTPC) is an applicable problem and a modified version of the MRCPSP in which all assumptions and constraints of the MRCPSP are held, but the objective function is different. We assume that there are a few renewable resources such as very expert human resources with high skill levels, particular types of cranes, and tunnel boring machines that should be leased from other companies providing these types of resources. Since these limited renewable resources are employed in other projects, there is a dictated ready date as well as a due date for each of them such that no resource can be accessible before its ready date, but these resources are allowed to be used after their due dates by paying penalty cost, depending on the resource type. Also, we suppose that there are a few nonrenewable resources like budget, materials, energy, or other resources which are consumed during the project.

Ranjbar et al. [2] studied this problem with single mode for each activity and availability of one unit for each type of renewable resource, without considering nonrenewable resources. They called this problem *resource-constrained project scheduling problem, minimization of total weighted resource tardiness penalty cost* (RCPSP-TWRTPC), which is an extended form of *resource-constrained project scheduling problem* (RCPSP). They developed a metaheuristic-based GRASP algorithm together with a branch and bound procedure to solve the problem.

The problem we have studied here is a generalization of the problem introduced by Ranjbar et al. [2] with more realistic viewpoint of resource costs by considering both renewable and nonrenewable resources cost. We call this problem *multimode resource-constrained project scheduling problem, minimization of total weighted resource tardiness penalty cost* (MRCPSP-TWRTPC).

Several exact and heuristic methods have been presented for MRCPSP. For instance, we can point to branch and cut method introduced by Heilmann [3] and branch and bound method developed by Zhu et al. [4] as two of the most powerful exact methods. Zhang et al. [5] presented classical particle swarm optimization (PSO) methods for multimode resource-constrained project scheduling problems to minimize the duration of construction projects. Lova et al. [6] suggested a heuristic algorithm based on priority rules and thereafter a hybrid genetic algorithm [7] to solve MRCPSP. Jarboui et al. [8] applied a combinatorial particle swarm optimization for solving multimode resource-constrained project scheduling problem. Ranjbar et al. [9] developed scatter search algorithm to tackle this problem, using path relinking methodology as its local search method. Van Peteghem and Vanhoucke [10] proposed genetic algorithm to solve preemptive and nonpreemptive multimode resource-constrained project scheduling problems. Kazemi and Tavakkoli-Moghaddam [11] developed a multimode particle swarm optimization which combines with genetic operator to solve a biobjective multimode resource-constrained project scheduling problem with positive and negative cash flows. The difference between this study and the previous papers is that they all considered the minimization of project makespan as objective function, but, in our problem, the objective function is minimization of total costs of renewable and nonrenewable resources.

MRCPSP-TWRTPC is a generalization of the RCPSP problem, and, considering the NP-hardness of RCPSP [12, 13], the MRCPSP-TWRTPC problem is NP-hard as well, and hence, metaheuristic method is the practical approach. In the remainder of this paper, we introduce a metaheuristic-based PSO algorithm for solving this problem. PSO, in its present form, dates back to 1990s; however, in this short period, PSO has shown good performance in a variety of application domains, particularly in the constrained optimization problems. Many researchers studied PSO widely and proposed several modifications. In this paper, we use a modified PSO algorithm developed and used by Tchomté and Gourgand for solving RCPSP efficiently [14].

The rest of this paper is organized as follows. In the next section, MRCPSP-TWRTPC is described in detail and is formulated in a mathematical model. In Section 3, a description of the PSO algorithm and its modifications is presented, and in Section 4 an algorithm based on modified PSO introduced by Tchomté and Gourgand [14] is explained. Section 5 is the experimental analysis. Finally, Section 6 concludes the work.

2. Problem Description

In MRCPSP-TWRTPC, a project is to be scheduled in order to minimize its total costs. Resources available for completing project activities can be classified as either renewable or nonrenewable. Activity j may have a number of execution modes M_j. Each activity mode specifies the activity duration and the activity requirements for the certain amount of renewable and nonrenewable resources. Each type of limited renewable resource is rented for a fixed time interval, starting from its ready time and ending with its due date, and is not available before its ready time but can be used after its due date with tardiness penalty cost. Nonrenewable resources are not limited. All activities are ready at the beginning of the project, and no preemption is permitted. If an activity is started under a specific mode, the activity mode cannot be changed. Activity j executed in mode m has duration d_{jm} and requires r_{jmk} units of renewable resource k and n_{jmk} units of nonrenewable resource k. The project network is depicted by an activity on node (AON) representation with finish-to-start precedence relations and zero time lag. Dummy activities 1 and n correspond to start and completion of the project. The list of activities is topologically numbered; that is, each predecessor of every activity has a smaller number than the number of activity itself. Also, we define the earliest and latest start time of activity j by EST_j and LST_j, respectively. EST_js and LST_js are computed by CPM forward and backward passes using the mode with shortest duration for each activity and assigning $\mathrm{LST}_n = \mathrm{LFT}_n = T$, where T is an upper bound for project makespan determined by any valid method, such as the sum of the longest duration of entire project activities plus the ready times of renewable resources. Consequently, each activity j can only be performed in time period $[\mathrm{EST}_j, \mathrm{LST}_j]$.

We define problem parameters as follows:

n: number of project activities,

NR: number of nonrenewable resources,

c_k: unit cost of nonrenewable resource k,

R: number of renewable resources,

R_k: renewable resource k availability,

r_k: ready time of renewable resource k,

d_k: due date of renewable resource k,

p_k: tardiness penalty cost of renewable resource k for each period,

M_j: number of modes of activity j,

P_j: the set of predecessors of activity j,

d_{jm}: duration of activity j under mode m,

r_{jmk}: renewable resource k requirement for executing activity j under mode m,

n_{jmk}: nonrenewable resource k requirement for executing activity j under mode m,

EST_j: earliest start time of activity j,

LST_j: latest start time of activity j,

T: upper bound of the project makespan.

We also define the decision variables as follows:

$$x_{jm\tau} = \begin{cases} 1, & \text{if activity } j \text{ is started under mode } m \text{ in period } \tau, \\ 0, & \text{otherwise,} \end{cases}$$
$$y_{k\tau} = \begin{cases} 1, & \text{if renewable resource } k \text{ is used in period } \tau, \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

l_k: is the renewable resource k tardiness, determined by $l_k = \max\{0,\ CP_k - d_k\}$, where CP_k is the release time of resource k by the project.

The mixed integer programming model for this problem can be formulated as follows:

$$\text{Min} \sum_{k=1}^{NR} c_k \left(\sum_{j=1}^{n} \sum_{m=1}^{M_j} n_{jmk} \sum_{\tau=\mathrm{EST}_j}^{\mathrm{LST}_j} x_{jm\tau} \right) + \sum_{k=1}^{R} p_k \cdot l_k \tag{2.2}$$

$$\text{S.t.} \sum_{m=1}^{M_j} \sum_{\tau=\mathrm{EST}_j}^{\mathrm{LST}_j} x_{jm\tau} = 1, \quad j = 1, 2, \ldots, n, \tag{2.3}$$

$$\sum_{m=1}^{M_i}\sum_{\tau=\mathrm{EST}_i}^{\mathrm{LST}_i}(\tau+d_{im})x_{imt} \le \sum_{m=1}^{M_j}\sum_{\tau=\mathrm{EST}_j}^{\mathrm{LST}_j}\tau x_{jmt}, \quad j=1,2,\ldots,n,\ i\in P_j, \tag{2.4}$$

$$\sum_{j=1}^{n}\sum_{m=1}^{M_j}r_{jmk}\sum_{z=\tau-d_{jm}+1}^{\tau}x_{jmz} \le R_k\cdot y_{k\tau}, \quad k=1,2,\ldots,R,\ \tau=1,2,\ldots,T, \tag{2.5}$$

$$\sum_{\tau=1}^{r_k-1}y_{k\tau}=0, \quad k=1,2,\ldots,R, \tag{2.6}$$

$$\tau\cdot y_{k\tau}-d_k\le l_k, \quad k=1,2,\ldots,R,\ \tau=d_k,\ d_k+1,\ldots,\mathrm{LST}_n, \tag{2.7}$$

$$x_{jm\tau}\in\{0,1\}, \quad j=1,2,\ldots,n,\ m=1,2,\ldots,M_j,\ \tau=\mathrm{EST}_j,\ldots,\mathrm{LST}_j, \tag{2.8}$$

$$y_{k\tau}\in\{0,1\}, \quad k=1,2,\ldots,R,\ \tau=0,\ldots,\mathrm{LST}_n, \tag{2.9}$$

$$l_k\ge 0, \quad k=1,2,\ldots,R. \tag{2.10}$$

In the above model, objective function (2.2) is project cost minimization in which the first and second terms are total costs of using nonrenewable resources and total penalty costs of renewable resources tardiness, respectively. Constraint set (2.3) ensures that each activity j is started under one of its modes within its specified start time periods, that is, $[\mathrm{EST}_j, \mathrm{LST}_j]$. Constraint set (2.4) forces precedence relationship between activities. Constrain (2.5) limit renewable resource usage. According to constraint (2.6), renewable resources cannot be used before their ready times and their tardiness periods are determined by constraint (2.7). Finally, constraint sets (2.8), (2.9), and (2.10) are nonfunctional ones.

3. Particle Swarm Optimization

The PSO algorithm is relatively recent, evolutionary, and population-based metaheuristic, originally developed by Kennedy and Eberhart [15] and redefined by Shi and Eberhart [16]. In spite of the early stages of the PSO method, it has found broad applications in combinatorial and constrained optimization domains and is currently a main research topic. PSO inspired from the social behavior of natural swarms exploits a swarm of particles for search space that are updated from iteration to iteration. Each particle corresponds to a candidate solution assessed by the objective function of the problem in question and is considered as a point in an n-dimension space. The status of a particle is represented by its position and velocity [15]. The n-dimensional position for particle i at iteration t can be represented as $X_{i,t}=(x_{i1}^t, x_{i2}^t,\ldots,x_{in}^t)$. Similarly, the velocity is also an n-dimension vector for particle i at iteration t which can be denoted as $V_{i,t}=(v_{i1}^t, v_{i2}^t,\ldots,v_{in}^t)$. Using the fitness evaluation, each particle remembers the best position it has perceived so far, referred to as $P_{i,t}$, and the best position of all particles, the best position of the best particle in the entire swarm, referred to as G_t. The position vector of each particle at iteration t is updated using (3.1), and the particle moves to its new position:

$$\mathrm{X}_{i,t+1}=X_{i,t}+V_{i,t}. \tag{3.1}$$

Velocity vector is also updated by(3.2)

$$V_{i,t+1} = c_1 \cdot V_{i,t} + c_2 \cdot r_2 \cdot (P_{i,t} - X_{i,t}) + c_3 \cdot r_3 \cdot (G_t - X_{i,t}). \tag{3.2}$$

This vector is a function of three components: previous velocity of the particle, the best experience of the particle, also, the entire swarm's best experiences up to the current iteration which are called inertia, cognition part, and social part, respectively [16].

Updating process continues until the termination criterion is met which usually is the maximal number of generations, processing time, or the best particle position of the whole swarm that cannot improve further after a predefined number of generations.

In (3.2), r_2 and r_3 are real random numbers with uniform distribution which are usually selected from the interval $[0, 1]$. c_2 and c_3 are known constants as learning factors, showing the significance of local and global best experiences, respectively. Also, c_1 is defined as a positive inertia weight which was first introduced by Shi and Eberhart [17]. This parameter can be specific for each particle [18]. Liu et al. [19] and Shi and Eberhart [20] introduced time-decreasing inertia weight.

The PSO parameters analyses have been the subject of several researches. For instance, Tchomté and Gourgand [14] determined some conditions for parameters to ensure that each particle converges to some equilibrium point after enough number of iterations.

Although PSO has been originally designed for solving continuous problems, it can be used for solving discrete problems as well. Different techniques have been designed to use this algorithm for combinatorial optimization problems such as the ones introduced by Jarboui et al. [8], Clerc [21], and Kennedy and Eberhart [22].

In this paper, we use the modified PSO approach which was introduced by Tchomté and Gourgand [14] as an extension of PSO that integrates a new displacement rule of the particles. The computational results of their algorithm showed that their PSO algorithm outperformed all state-of-the-art algorithms in solving RCPSP, and this is the reason for selecting their approach for our problem. We describe this modified PSO method in the following.

A metaheuristic algorithm should be able to explore search space effectively and efficiently. PSO algorithm should be intelligent enough to both intensively explore regions of the search space with high-quality solutions and to diversely move to unexplored regions of the search space. These two techniques that were introduced by Glover and Laguna [23] are known as intensification and diversification methods, respectively. Tchomté and Gourgand [14] analyzed particle trajectories and modified particle position updating rules. The idea originated from the PSO applications in which particles basically move from their current positions toward the best local and global positions $(P_{i,t}, G_t)$, but the particles do not get close enough to $P_{i,t}$ and G_t. As a result, diversification is performed well, but intensification is not. Consequently, Tchomté and Gourgand [14] proposed a new particle displacement rule to improve the intensification process by letting each particle visit two positions $S_{i,t}$ and $T_{i,t}$ before moving from current position, $X_{i,t}$, to the next position $X_{i,t+1}$. First, the inertia has influences on the position by making the particle move from $X_{i,t}$ to $S_{i,t}$. Then the cognition part moves the particle to $T_{i,t}$ and finally under social part affect the particle to reach its new position, $X_{i,t+1}$, at the next iteration. Adapted from Tchomté and Gourgand [14], particle displacement in the classical PSO and this modified PSO has been shown in Figures 1(a) and 1(b), respectively.

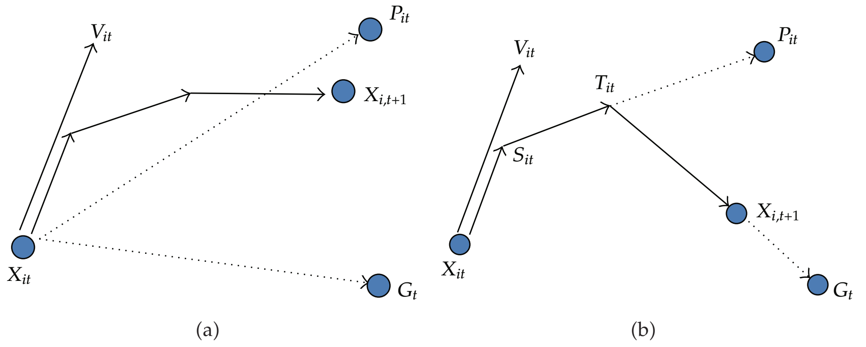


Figure 1: Particle displacement in the swarm: (a) classical PSO, (b) modified PSO.

For this purpose, the particle position updating rule is

$$\begin{aligned} S_{i,t} &= X_{i,t} + c_1 \cdot V_{i,t}, \\ T_{i,t} &= S_{i,t} + c_2 \cdot r_2 \cdot (P_{i,t} - S_{i,t}), \\ X_{i,t+1} &= T_{i,t} + c_3 \cdot r_3 \cdot (G_t - T_{i,t}), \\ X_{i,t+1} &= X_{i,t} + c_1 \cdot (1 - c_2 \cdot r_2) \cdot (1 - c_3 \cdot r_3) \cdot V_{i,t} \\ &\quad + c_2 \cdot r_2 \cdot (1 - c_3 \cdot r_3) \cdot (P_{i,t} - X_{i,t}) + c_3 \cdot r_3 \cdot (G_t - X_{i,t}). \end{aligned} \tag{3.3}$$

Then, in each time step t, velocity vector is updated as follows:

$$V_{i,t+1} = \alpha \cdot V_{i,t} + \beta \cdot (P_{i,t} - X_{i,t}) + \gamma \cdot (G_t - X_{i,t}), \tag{3.4}$$

$$\begin{aligned} \alpha &= c_1 \cdot (1 - r_2 \cdot c_2) \cdot (1 - r_3 \cdot c_3), \\ \beta &= c_2 \cdot (1 - r_3 \cdot c_3), \\ \gamma &= c_3. \end{aligned} \tag{3.5}$$

Tchomté and Gourgand [14] showed that the necessary conditions for coefficients so that the particles converge to the equilibriums are satisfying (3.6) plus (3.7) or (3.6) plus (3.8):

$$\phi > 0, \qquad \phi - 2 * (c_1 + 1) < 0, \qquad c_1 < 1, \tag{3.6}$$

where $\phi = (c_2 + c_3)/2$,

$$0 < c_1 < 0.9, \qquad 0 < c_2 < 2, \qquad 0 < c_3 < 2, \tag{3.7}$$

$$0 < c_1 < 0.9, \qquad 2 \le c_2 < 4, \qquad 2 \le c_3 < 4. \tag{3.8}$$

4. Modified PSO for MRCPSP-TWRTPC

In this section, we present a modified PSO algorithm, using the approach of Tchomté and Gourgand [14], for solving MRCPSP-TWRTPC. Algorithm 1 shows the pseudocode. In this algorithm the ith particle position at iteration t is represented by the n-dimensional vector

```
(1) Do Preprocessing
(2) Generate initial particle swarm
(3) While termination criterion is met do
(4)     While all particles have been evaluated do
(5)         Determine activities priorities
(6)         Schedule activities based on their modes and priorities using the parallel
              schedule generation and delay local search
(7)         While schedule is improved do
(8)             Improve schedule by Mode Assignment Modification—Part I
(9)             Improve schedule by Local Left Shift
(10)        End while
(11)        Improve schedule by Mode Assignment Modification—Part II
(12)        Compute corresponding cost of the generated schedule
(13)    End while
(14)    Update the local and global best solutions if necessary
(15)    Update position and velocity of each particle according to (3.3) and (3.4), respectively
(16) End while
(17) Report the global best solution
```

Algorithm 1: Pseudocode of modified PSO algorithm for MRCPSP-TWRTPC.

$X_{i,t} = (x^t_{i1},\ x^t_{i2}, \ldots,\ x^t_{in})$ in which $x^t_{ij},\ j = 1, 2, 3, \ldots, n$ is the mode assignment to the jth activity at iteration t and is an integer in the interval $[1, M_j]$. A feasible schedule of the project is constructed from each $X_{i,t}$. For this purpose, first the activities are prioritized, see Section 4.3. Then, using single-pass parallel schedule generation scheme, the activities are scheduled, see Section 4.4. Certain local search and improvement procedures are applied on this solution to reach a better schedule, see Sections 4.5, 4.6, 4.7, and 4.8.

Each particle's fitness value is determined by calculating total cost of the final schedule. Then, if necessary, local and/or global best positions are updated, see Section 4.9. If termination criterion is not met, particle positions and velocity vectors are updated by (3.3) and (3.4), respectively, for the next iteration, see Section 4.10. Different parts of this algorithm are described in more details as follows.

4.1. Preprocessing

Sprecher et al. [24] introduced several preprocessing rules in order to reduce feasible space of MRCPSP. Later, these rules have been used in other articles such as Lova et al. [6], Peteghem and Van Vanhouck [10], and Hartmann and Briskorn [25]. Considering the similarities between MRCPSP-TWRTPC and MRCPSP, we apply two of these rules to our proposed problem. One is the *nonexecutable mode elimination* rule for an activity. For a *nonexecutable mode,* the amount of the resource needed for executing the activity is more than the resource availability. Another method is *inefficient mode elimination* method. A given mode is *inefficient* for an activity if there is another mode for which the activity duration is less, and that activity can be accomplished with less total amount of both renewable and nonrenewable resources. Therefore, activities modes are analyzed one by one and nonexecutable and inefficient modes are deleted.

4.2. Generating Initial Particle Swarm

Initial particle swarm is generated randomly. Here, component $j,\ j = 1, \ldots, n$, of either position or velocity vectors for each particle is generated randomly from the ranges $[1, M_j]$ and

$[-M_j, M_j]$, respectively, as there is no nonrenewable resource constraint in the problem; moreover, all nonexecutable modes have been deleted, initial mode assignments are feasible, and no modification is needed.

4.3. Activity Priority for Scheduling

In order to generate a solution in MRCPSP-TWRTPC, two issues are to be decided: activities mode assignment and scheduling of activities. By specifying mode assignment for a solution, the cost of nonrenewable resources is determined and fixed. Then, scheduling of activities is performed with the objective of minimizing the total cost of renewable resource tardiness penalties. Therefore, the priorities of activities for scheduling are determined by renewable resources. Our procedure for this purpose is as follows.

First, we define the set of activities which need renewable resource k as the *activity set of resource k* (ASR_k). Each activity in this set may have immediate or nonimmediate predecessors that may not be a member of this set. We define the set of these predecessors which are not members of ASR_k as *activities predecessors of resource k* (APR_k). Then, the pairs of ASR_k and APR_k, $k = 1, \ldots, R$, are prioritized by index k using the heuristic that activities in ASR_k and APR_k for the resource which has more potential of causing tardiness penalty should receive higher priority of being scheduled. To access the potential of the kth resource tardiness penalty cost, we note that this penalty cost is equal to $P_k \times \max\{0, CP_k - d_k\}$, where CP_k is the release time of resource k in the project, and hence $P_k \times (CP_k - d_k)$ is a good measure for prioritizing the resources for this purpose. Now the question is how to access CP_k without knowing the schedule. We use the following procedure to access CP_k, $k = 1, \ldots, R$.

Since no activity can start sooner than the ready time of all the resources it needs, in order to take into account the resource ready time, we add one dummy node k for each resource k, $k = 1, \ldots, R$, to the project network. Each dummy activity R is single mode with no resource requirement, and the duration of r_k. All these dummy activities can be scheduled at time zero. Then, for any activity j which needs renewable resource k, we set dummy activity k as one of its predecessors. So execution of activity j is not possible before the time r_k. Note that dummy activity k is a member of APR_k. The length of the critical path of subnetwork k is denoted by CP_k and is considered as a relative measure of CP_k for $k = 1, \ldots, R$. CP_k is computed using CPM method. After computation of CP_k for all resources, the parts of activity sets ASR_k and APR_k, $k = 1, \ldots, R$, are prioritizing using the value of $P_k \times (CP_k - d_k)$.

In our procedure for prioritizing activities for scheduling, we give more priority to the activities in APR_k than activities in ASR_k, since activities in ASR_k cannot be scheduled unless activities in APR_k are scheduled.

Finally, for each resource k, it is necessary to prioritize the activities belonging to each set of ASR_k and APR_k. We notice when a set of activities using a renewable resource are to be scheduled, we actually deal with an RCPSP, because activities under specified modes are to be scheduled in order to execute within the shortest possible time. Therefore, in order to prioritize activities of a set of ASR_k or APR_k, we apply one of the most efficient heuristics to scheduling of activities in RCPSP. Lova et al. [6] compared a number of most efficient heuristics for prioritization of activities in RCPSP and found out that activities prioritization based on nondecreasing order of the sum of the latest start and finish time (LSTLFT) gained the best results among single-pass heuristics. This has been among the best multipass methods as well. Multipass methods need much more computation times than single-pass methods, but they usually result in negligible improvement of the solution; hence, we use the LSTLFT method with single-pass scheduling in order to prioritize activities of ASR_k and APR_k.

4.4. Scheduling Activities

We use parallel schedule generation scheme for scheduling activities, since it fits well with *delay local search* which we will use. For more details on parallel schedule generation scheme see Section 6.3.2.1.2 of Demeulemeester and Herroelen [1].

Parallel schedule generation scheme repeats over the separate decision points at which activities can be added to the schedule. A decision point in time horizon corresponds with the completion times of already scheduled activities; thus, at most n decision points need to be considered. At each decision point, the unscheduled activities whose predecessors have been accomplished are taken into consideration in the order of priority list and are scheduled if no resource conflict exists at that time instant.

In this problem, renewable resources have ready time and availability of them differs before and after these times and some new activities may be able to be scheduled after these times. Therefore, we consider ready times in addition to the completion times of activities for choosing new point in time horizon. A new point in time horizon is the closest point to the current point among the ready times of renewable resources and the completion time of scheduled activities.

4.5. Delay Local Search

This local search method was used by Chen et al. [26] for the RCPSP problem to escape from local minimum. This method performs like the mutation operator in genetic algorithm and can delay scheduling each activity in spite of its priority to let other activities be scheduled sooner and some resources asked by selected activity will be retained for other activities.

In order to use resources more efficiently, scheduling some activities are delayed in spite of their priority. So other activities can be scheduled sooner. If these selected activities are not delayed, they could delay other project activities for a rather long time. Therefore, each activity is delayed if $q \leq q_0$, where q is randomly selected form uniform distribution $[0, 1]$ and q_0 $(0 < q_0 < 1)$ is the probability of delay and called "delay rate."

Considering the efficiency of this delay local search in shortening project makespan shown in Chen et al. [26], we apply this method to scheduling activities.

4.6. Mode Assignment Modification-Part I

After scheduling activities, the current schedule may have a set of activities with positive free floats. We call this set FFA. For each $j \in FFA$, it may be possible to change the mode of activity j and reschedule it, using its free float, such that its finish time is not increased and hence no other activity is affected. The mode change and rescheduling of activity j can reduce nonrenewable resource costs, but it can also change CP_k, release time of resource k, $k = 1, \ldots, R$. The change of CP_k can change renewable resource k tardiness, $k = \max\{0,\ CP_k - d_k\}$, and its cost, $p_k l_k$. If we set the availability of resource k, $k = 1, \ldots, R$, for the periods after $\max\{CP_k, d_k\}$ equal to zero and then reschedule activity j, we are sure that this scheduling is not going to increase renewable resource tardiness penalties. Considering the above points, we define "*mode assignment modification-part I*" as follows and use it as a local search procedure in our algorithm.

(1) For the current schedule find the set FFA by forward pass computation.

(2) Set the availability of resource k, $k = 1, \ldots, R$, for the periods after $\max\{CP_k, d_k\}$ equal to zero.

(3) For each $j \in$ FFA, as we go through forward pass, consider the least nonrenewable resource cost mode of activity j. If it is not its current mode, reschedule activity j in this mode using its free float and considering resource constraints, without increasing its finish time. If this rescheduling is not possible, check the next least nonrenewable resource cost mode of activity j.

4.7. Local Left Shift Improvement

Mode assignment modification-part I can reduce the renewable resource requirements of the project for certain periods under the resulted schedule. This should be expected since usually the mode with less nonrenewable resources has longer duration and requires less renewable resources per period. Hence, the possibility of local left shift of certain activities exists. Therefore, we perform the standard local left shift method after the mode assignment modification-part I.

After performing local left shift, we might be able to modify some of the activities mode assignment again by mode assignment modification-part I. Hence, these two procedures are used one after another until no improvement can be achieved in the schedule.

4.8. Mode Assignment Modification-Part II

We consider the set of project activities with no successors and call this set NSA. The direct predecessor activities of dummy activity n make NSA. For any $j \in$ NSA if we change the mode of activity j and reschedule it, the schedule of no other activity changes and the value of cost change of the project is $\Delta c = \Delta \mathrm{NRC}_j + \sum_{k=1}^{R} p_k \Delta l_k$, where $\Delta \mathrm{NRC}_j$ is the change of nonrenewable resource cost of mode for activity j and Δl_k is the change of kth resource tardiness. Since activity j has no successor, Δl_k can be computed easily. If the value of change effect on total cost is negative, the mode change for activity $j \in$ NSA is justifiable. Considering above points, we define the following local search procedure as "*mode assignment modification-part II*" and use it in our algorithm.

(1) Let $NSA = \{j \mid \text{activity } j \text{ is the direct predecessor of dummy activity } n\}$.

(2) For each $j \in NSA$, consider some modes for activity j, in which nonrenewable resources requirement cost is less and this cost saving is more than the probable increase in the penalty cost of renewable resources. Compute Δc for each of them. Considering renewable resource constraints, if the least Δc is negative, its corresponding mode replaces the current mode of activity j.

4.9. Updating the Local and Global Best Solutions

As mentioned earlier, the PSO algorithm stores the best local solution obtained by each particle and the best global solution obtained by the entire swarm. Therefore, after evaluating all the particles of the swarm at iteration, the best local solution for each particle up to the current iteration is compared with the current solution of the particle. If the current solution has lower total cost, the best local solution of the particle is replaced by the current solution of the particle.

Similarly, the best global solution is updated.

4.10. Updating Particles Position and Velocity

To update position of the particles to generate new solutions for the next iteration, firstly, particles velocity is updated using (3.4). In this process each element of the velocity vector which is more than M_i is changed to M_i and each element of the velocity vector which is less than $-M_i$ is changed to $-M_i$. Then, particles positions are updated using (3.3). Similar to the procedure for updating each particle velocity, each element of the new position vector which is more than M_i is changed to M_i and each element of the new position vector which is less than 1 is changed to 1. As the elements of position vector determine the mode assignment of activities and should be integer, each noninteger element is rounded to the closest integer.

5. Experimental Analysis

In this section, we present experimental analysis of the algorithm. All programs have been coded and executed on C#.NET 2008 platform on a PC with Core 2 Duo 2.53 GHz CPU and 3 GB RAM.

5.1. Sample Problems

We used sample problems library of PSPLIB [27] and selected three sets of multimode project scheduling problems, j10, j16, j20 as the small size problems and j30 as the medium size problem. In addition, two sets of large size problems, j60 and j90, were generated with the same parameters as j30, but with 60 and 90 activities, respectively. Also, in order to observe the effect of resources in the problem, two extra sets of project scheduling problems were generated by Kolisch et al. [28], which we call j30_r4_n4 and j60_r4_n4. All the parameters in these sets are similar to those in sets j30 and j60, respectively, but instead of having 2 renewable and 2 nonrenewable resources, there exist 4 resources of each type.

Discrete uniform distribution has been used in the related literature of the project scheduling; for example, Ranjbar et al. [2] and Khalilzadeh et al. [29] used discrete uniform distribution for resource ready dates and due date. Hence, in this paper we have used discrete uniform distribution to select the parameters. The unit cost of nonrenewable resources were randomly selected from discrete uniform distribution (2,6), the unit penalty cost of renewable resource tardiness were randomly chosen from discrete uniform distribution (10,30). The ready times of renewable resources were randomly generated from discrete uniform distribution (0,15), and finally, the renewable resource due dates were randomly picked from discrete uniform distribution (5,15) plus the amount of their ready time.

5.2. Parameters Tuning

There exist five parameters, delay rate (q_0), the number of particles in the swarm P, and also, c_1, c_2, and c_3 in our PSO algorithm. For setting these parameters we used 3-point factorial design as shown in Table 1. As mentioned in Section 3, parameters c_1, c_2, and c_3 ought to be chosen in the ranges given by (3.6) and (3.7), or (3.6) and (3.8). For each of these two ranges, three points have been selected.

A set of 100 test instances was randomly selected from the set j16 and solved with the CPU time limit of 150 milliseconds. Subsequently, around 4 to 30 iterations were executed, depending on the number of particles in the swarm.

Table 1: Parameter settings.

Parameter	Levels	
q_0	0.2–0.4–0.6	
P	10–30–60	
c_1	0.2–0.45–0.7	
	Satisfying relations (3.6) and (3.7)	Satisfying relations (3.6) and (3.8)
c_2	0.5–1–1.5	2.5–3–3.5
c_3	0.5–1–1.5	2.5–3–3.5

Table 2: Algorithm validity assessment.

Problems set	Average CPU time (millisecond)	Average d	Standard deviation of d
J10 (536 problems)	278	1.19	2.18
J16 (550 problems)	446	3.64	4.75
J20 (554 problems)	543	4.70	5.65
J30 (640 problems)	825	7.40	8.13
J60 (640 problems)	1818	10.56	11.06
J90 (640 problems)	2916	11.76	12.21
J30-r4-n4 (640 problems)	1509	7.78	8.73
J60-r4-n4 (640 problems)	3302	11.69	12.56

Each test instance was run for all $3^5 \times 2 = 486$ permutations of parameter values, a total of 48,600 problems. The tuned values of the parameters are $q_0 = 0.4$, $P = 60$, $c_1 = 0.2$, $c_2 = 0.5$, $c_3 = 1$.

5.3. Algorithm Validity

As our research is new, we could not find any solved problem in the literature. So we developed some instance problems whose optimum objective function values are in hand. Then we solved these instances with our algorithm and tested the results. In order to generate these instances, we used sample problems introduced in Section 5.1; however, we modified the due dates of renewable resources as follows. We generated a random feasible schedule for each instance, after assigning the least nonrenewable cost mode to each activity of the project. In this schedule, we determined the release time of each renewable resource. Subsequently, we set the due date for each renewable resource equal to its release time; hence, this schedule has zero tardiness penalty cost and its objective function value is equal to the cost of nonrenewable resources. As we assigned the least nonrenewable cost modes to the activities, this schedule is optimal and the optimum value of its objective function is available.

All sample problems were modified with the above procedure and solved with the PSO algorithm. The termination criterion was generation of 600 schedules. In order to assess the validity of the algorithm, d, the percent deviation of the objective function value from optimum, computed for each test problem solved, where $d = 100 \times (Z_p - Z_{\text{opt}})/Z_{\text{opt}}$, Z_p is the objective function value of the best solution achieved by the PSO algorithm and Z_{opt}, the optimal objective function value of the instance. Table 2 shows the average and standard deviation of d for each problem set. The low values of average and standard deviation of d reveal good performance of the algorithm with small CPU time, although we do not have any standard for comparison.

Table 3: Algorithm robustness check.

Problem set	120 schedules		900 schedules	
	Average CPU time (millisecond)	Average percent deviation	Average CPU time (millisecond)	Average percent deviation
j10	69	6.18	352	5.17
j16	112	4.74	541	5.01
j20	135	4.14	648	4.14
J30	216	4.17	1039	3.65
J60	471	3.44	2220	3.35
J90	776	2.13	3652	2.08
J30-r4-n4	410	4.13	1893	3.29
J60-r4-n4	880	2.77	4085	2.76

5.4. Algorithm Robustness

In order to check the robustness of the PSO algorithm, we have used the algorithm for several instances. Each instance has been solved several times, and d the percent deviation of objective function value for each instance has been computed.

From each set of problems, 15 randomly chosen instances have been used and each instance has been solved 30 times using the PSO algorithm, and each time done under two termination criteria, generation of 120 and 900 schedules.

To check the robustness of the algorithm, d' for each instance has been computed, where $d' = 100 \times \mathrm{Sd}(Z_i)/E(Z_i)$, Z_i is the objective function value of the best solution achieved by solving the problem for ith run, $\mathrm{Sd}(Z_i)$ standard deviation of Z_i, and $E(Z_i)$ is the mean of Z_i. The average of d' for the problems of each set has been shown in Table 3.

5.5. Improvement Methods Performance Assessment

In this section, we assess the performance of the improvement methods introduced in Sections 4.6 and 4.8. The first one is the mode assignment improvement method-part I along with local left shift, and the other one is the mode assignment improvement method-part II. In order to assess each of these improvement methods, we remove each one from the original algorithm to get two simplified algorithms, each of which does not have one of the two improvement methods. We use 30 instances from each problem set and solve them with the main algorithm, also, with the two simplified algorithms. Subsequently, compare the results gained by simplified algorithms with the results of the original algorithm. All instances have been solved by three algorithms under three different termination criteria, to generation of 120, 600, and 900 schedules. We compute d'' for each instance to compare the results, where $d'' = 100 \times (Z_s - Z_p)/Z_p$, Z_s is the final solution objective function value obtained by the simplified PSO algorithm, Z_p, final solution objective function value obtained by original PSO, and d'' is the percent deviation of Z_s from Z_p.

Mean and standard deviation of d'' for each set have been computed. Table 4 shows the effect of deleting mode assignment improvement method-part I and local left shift and we can see that, in all cases, the performance of the algorithm deteriorates remarkably and in most cases the mean CPU time, in millisecond, considerably increases if this local search is deleted.

Table 4: Performance assessment of improvement part I.

Problem set	120 schedules				600 schedules				900 schedules			
	Average CPU time of original algorithm (millisecond)	Average CPU time of simplified algorithm (millisecond)	Average d	Standard deviation of d	Average CPU time of original algorithm (millisecond)	Average CPU time of simplified algorithm (millisecond)	Average d	Standard deviation of d	Average CPU time of original algorithm (millisecond)	Average CPU time of simplified algorithm (millisecond)	Average d	Standard deviation of d
J10	87	85	6.43	14.13	270	335	7.27	15.77	385	487	15.90	27.79
J16	121	133	10.32	16.65	707	523	4.54	9.11	605	791	6.66	12.75
J20	157	171	10.61	15.71	515	654	10.43	17.49	725	983	12.63	21.84
J30	233	247	5.41	9.93	787	1005	8.58	14.75	1120	1488	10.41	14.92
J60	499	541	10.46	12.29	1658	2189	8.95	10.72	2419	3257	10.33	13.26
J90	819	886	10.05	11.57	2676	3531	11.23	11.83	3960	5293	11.82	13.04
J30-r4-n4	437	476	9.43	13.34	1423	1964	8.81	13.55	2082	2909	11.71	13.81
J60-r4-n4	887	1003	12.12	13.94	2950	7089	12.72	15.03	4294	6084	13.13	14.78

Table 5: Performance assessment of improvement part II.

	120 schedules				600 schedules				900 schedules			
Problem set	Average CPU time of original algorithm (millisecond)	Average CPU time of simplified algorithm (millisecond)	Average d	Standard deviation of d	Average CPU time of original algorithm (millisecond)	Average CPU time of simplified algorithm (millisecond)	Average d	Standard deviation of d	Average CPU time of original algorithm (millisecond)	Average CPU time of simplified algorithm (millisecond)	Average d	Standard deviation of d
J10	87	112	0.61	8.65	270	429	3.32	11.92	385	646	4.37	14.77
J16	121	175	2.45	7.62	707	676	5.47	10.20	605	1032	2.04	8.66
J20	157	220	1.50	7.41	515	842	0.38	5.85	725	1231	0.66	7.96
J30	233	341	−1.52	8.06	787	1313	1.68	5.29	1120	1922	2.16	6.63
J60	499	730	−0.54	3.19	1658	2858	−0.55	3.74	2419	4208	0.68	4.25
J90	819	1167	0.06	3.55	2676	4575	−0.73	4.32	3960	6752	−0.93	2.84
J30-r4-n4	437	618	−0.70	6.74	1423	2413	−0.04	3.61	2082	3611	0.39	4.27
J60-r4-n4	887	1322	0.77	3.89	2950	5102	1.34	4.70	4294	7505	−0.62	3.45

Table 5 shows the effect of deleting mode assignment improvement method-part II. We can see that in most cases the performance of the algorithm deteriorates, and in all cases the average CPU time increases remarkably. In the following, we explain the reason for increasing the average CPU time.

6. Conclusions

In this paper, we introduced MRCPSP-TWRTPC problem as a resource-oriented cost minimization project scheduling problem considering both renewable and nonrenewable resource costs. We formulated and mathematically modeled this problem as mixed integer programming model and discussed its NP-hardness. Subsequently, we developed a metaheuristic algorithm to tackle the proposed project scheduling problem. We briefly reviewed the applications of the PSO algorithm for solving combinatorial and constrained optimization problems. Thereafter, we applied a modified PSO algorithm including modified updating rules for particles velocity and position. In order to generate feasible schedules, we used the PSO algorithm for activity mode assignment and developed a novel heuristic technique to prioritize activities for parallel scheduling scheme. Two improvement heuristics, delay local search and local left shift, in line with two mode assignment modification methods, were implemented to improve the solutions. The computational results revealed proper algorithm robustness in solving different instances especially with high number of iterations. Also, the validity analysis showed small deviations from the optimal solutions for the test instances in reasonable solving time. Finally, we assessed two improvement methods used in our algorithm to demonstrate their good performance.

References

[1] E. Demeulemeester and W. S. Herroelen, *Project Scheduling, A Research Handbook*, Kluwer Academic, Dordrecht, The Netherlands, 2001.

[2] M. Ranjbar, M. Khalilzadeh, F. Kianfar, and K. Etminani, "An optimal procedure for minimizing total weighted resource tardiness penalty costs in the resource-constrained project scheduling problem," *Computers and Industrial Engineering*, vol. 62, no. 1, pp. 264–270, 2012.

[3] R. Heilmann, "A branch-and-bound procedure for the multi-mode resource-constrained project scheduling problem with minimum and maximum time lags," *European Journal of Operational Research*, vol. 144, no. 2, pp. 348–365, 2003.

[4] G. Zhu, J. F. Bard, and G. Yu, "A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem," *INFORMS Journal on Computing*, vol. 18, no. 3, pp. 377–390, 2006.

[5] 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.

[6] A. Lova, P. Tormos, and F. Barber, "Multi-mode resource constrained project scheduling: scheduling schemes, priority rules and mode selection rules," *Inteligencia Artificial*, vol. 10, no. 30, pp. 69–86, 2006.

[7] 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.

[8] 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.

[9] 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.

[10] V. Van 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.

[11] F. S. Kazemi and R. Tavakkoli-Moghaddam, "Solving a multi-objective multi-mode resource-constrained project scheduling problem with particle swarm optimization," *International Journal of Academic Research*, vol. 3, no. 1, pp. 103–110, 2011.

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

[13] P. Brucker, A. Drexel, R. Mohring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: notation, classification, models and methods," *European Journal of Operational Research*, vol. 113, pp. 3–41, 1999.

[14] S. Kemmoé Tchomté and M. Gourgand, "Particle swarm optimization: a study of particle displacement for solving continuous and combinatorial optimization problems," *International Journal of Production Economics*, vol. 121, no. 1, pp. 57–67, 2009.

[15] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942—1948, Piscataway, NJ, USA, December 1995.

[16] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, Piscataway, NJ, USA, May 1998.

[17] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Annual Conference Evolutionary Programming*, San Diego, Calif, USA, 1998.

[18] C. Y. Tsai and S. W. Yeh, "A multiple objective particle swarm optimization approach for inventory classification," *International Journal of Production Economics*, vol. 114, no. 2, pp. 656–666, 2008.

[19] S. Liu, J. Tang, and J. Song, "Order-planning model and algorithm for manufacturing steel sheets," *International Journal of Production Economics*, vol. 100, no. 1, pp. 30–43, 2006.

[20] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 945–1950, Piscataway, NJ, USA, 1999.

[21] M. Clerc, "Discrete particle swarm optimization," in *New Optimization Techniques in Engineering*, G. C. Onwubolu and B. V. Babu, Eds., pp. 204–219, Springer, Berlin, Germany, 2004.

[22] Kennedy and R. C. Eberhart, "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, IEEE Computer Society, Washington, DC, USA, 1997.

[23] F. Glover and M. Laguna, "Tabu search," in *Handbook of Combinatorial Optimization*, D.-Z. Du and P. M. Pardalos, Eds., vol. 3, pp. 621–757, Kluwer Academic, Boston, Mass, USA, 1998.

[24] A. Sprecher, S. Hartmann, and A. Drexl, "An exact algorithm for project scheduling with multiple modes," *OR Spektrum*, vol. 19, no. 3, pp. 195–203, 1997.

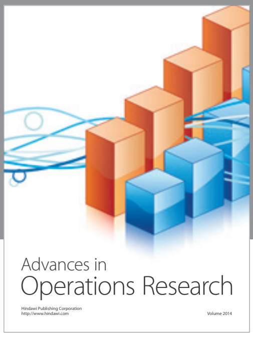
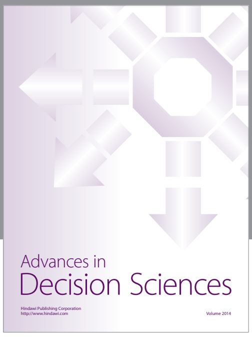
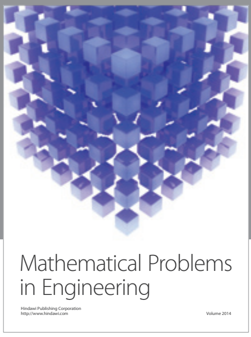
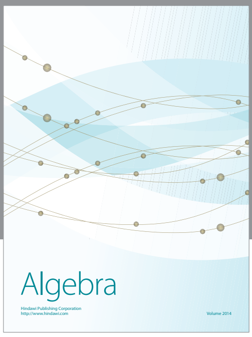
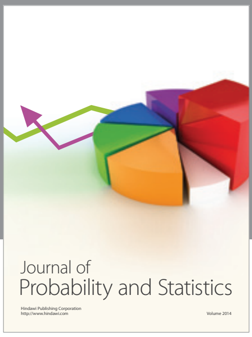
[25] S. Hartmann and D. Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 207, no. 1, pp. 1–14, 2010.

[26] 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.

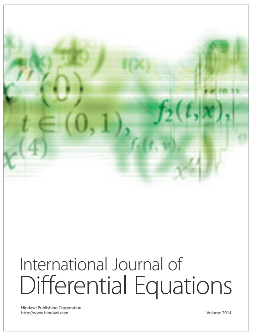
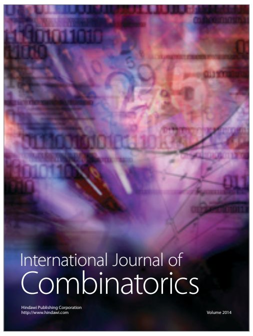
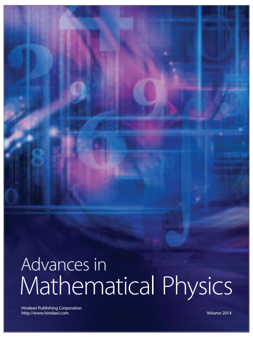
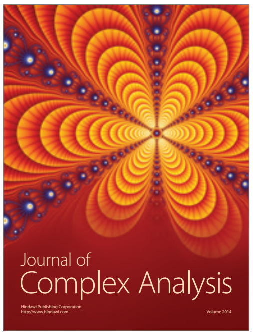
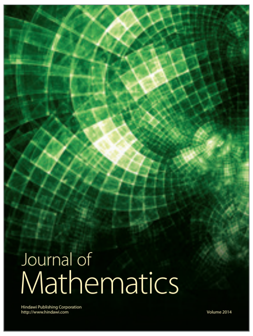
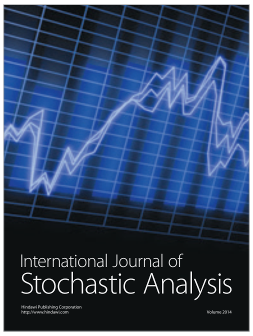
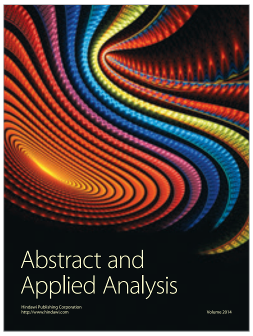
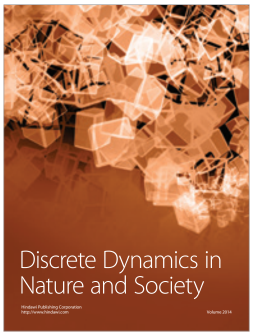
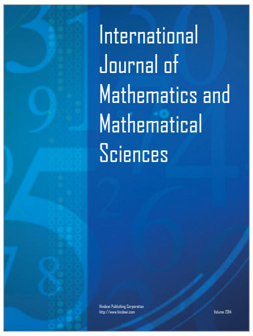
[27] R. Kolisch and A. Sprecher, "PSPLIB—a project scheduling problem library," *European Journal of Operational Research*, vol. 96, no. 1, pp. 205–216, 1997.

[28] R. Kolisch, A. Sprecher, and A. Drexl, "Characterization and generation of a general class of resource-constrained project scheduling problems," *Management Science*, vol. 41, pp. 693–1703, 1995.

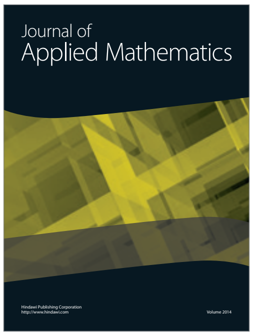
[29] M. Khalilzadeh, F. Kianfar, and M. Ranjbar, "A Scatter Search Algorithm for RCPSP with discounted weighted earliness-tardiness costs," *Life Science Journal*, vol. 8, no. 2, pp. 634–640, 2011.


Advances in
Operations Research

Advances in
Decision Sciences

Mathematical Problems
in Engineering

Algebra

Journal of
Probability and Statistics

The Scientific
World Journal


International Journal of
Differential Equations

International Journal of
Combinatorics

Advances in
Mathematical Physics

Journal of
Complex Analysis

Journal of
Mathematics

International Journal of
Stochastic Analysis

Abstract and
Applied Analysis

Discrete Dynamics in
Nature and Society

International
Journal of
Mathematics and
Mathematical
Sciences

Journal of
Discrete Mathematics

Journal of
Function Spaces

Journal of
Applied Mathematics

Journal of
Optimization