Hindawi Publishing Corporation
Advances in Operations Research
Volume 2014, Article ID 505716, 10 pages
http://dx.doi.org/10.1155/2014/505716

Research Article

Resource Constrained Project Scheduling Subject to Due Dates: Preemption Permitted with Penalty

Behrouz Afshar-Nadjafi

Faculty of Industrial and Mechanical Engineering, Islamic Azad University, Qazvin Branch, P.O. Box 34185-1416, Qazvin 34185-14161, Iran

Correspondence should be addressed to Behrouz Afshar-Nadjafi; afsharnb@alum.sharif.edu

Received 7 January 2014; Revised 27 March 2014; Accepted 2 April 2014; Published 16 April 2014

Academic Editor: Imed Kacem

Extensive research works have been carried out in resource constrained project scheduling problem. However, scarce researches have studied the problems in which a setup cost must be incurred if activities are preempted. In this research, we investigate the resource constrained project scheduling problem to minimize the total project cost, considering earliness-tardiness and preemption penalties. A mixed integer programming formulation is proposed for the problem. The resulting problem is NP-hard. So, we try to obtain a satisfying solution using simulated annealing (SA) algorithm. The efficiency of the proposed algorithm is tested based on 150 randomly produced examples. Statistical comparison in terms of the computational times and objective function indicates that the proposed algorithm is efficient and effective.

1. Introduction

Preemptive project scheduling problems are those in which the accomplishing of an activity is temporarily preempted and restarted afterwards. In the previous literature on preemptive project scheduling, preempted activities simply are resumed from the moment at which preemption occurred without any cost. However, this situation is not always true in reality. It is probable that, in some situations, a certain setup or delay cost should be considered.

The literature on solution approaches for the preemptive resource constrained project scheduling problem with weighted earliness-tardiness and preemption penalties (PRCPSP-WETPP) is scant. Of course, several papers have been devoted to machine scheduling considering preemption costs. Potts and van Wassenhove [1] integrated preemptive scheduling with batching and lot-sizing model. Monma and Potts [2] and Chen [3] suggested the heuristics for parallel machine scheduling problem subject to preemption and batch setup times. Zdrzalka [4], Schuurman and Woeginger [5], and Liu and Cheng [6] studied preemptive scheduling with job release dates and job-dependent setup times. Julien et al. [7] proposed generalized preemption models for single-machine dynamic scheduling problems. Rebai et al. [8] developed some metaheuristics and exact methods for minimization of earliness-tardiness penalties on a single machine to schedule preventive maintenance tasks. They used linear programming and branch and bound to obtain exact solutions. Also, they developed a local search approach as well as a genetic algorithm as metaheuristics for solving hard scheduling problems.

Vanhoucke [9] and Vanhoucke et al. [10] have proposed an exact method for the weighted earliness-tardiness project scheduling problem (WETPSP) in which activities are non-preemptive. Also, resources are assumed unbounded. The algorithm has two steps: first, it schedules the activities at their due date or later if forced so by precedence constraints. Then, a recursive procedure computes the optimal displacement for those activities that their left shift is beneficial. Vanhoucke et al. [11] developed a branch and bound (B&B) procedure to solve the WETPSP for maximizing the net present value (NPV) of a project subject to progress payments. Net present value (NPV) is the difference between the present value of cash inflows and the present value of cash outflows. NPV is a standard method for using the time value of money which is used in capital budgeting to analyze

the profitability of an investment or project. Vanhoucke et al. [11] have applied the logic of their recursive algorithm in the proposed B&B. Their branching strategy resolves resource conflicts through the addition of extra precedence relations based on the concept of minimal delaying alternatives. Afshar-Nadjafi and Shadrokh proposed a B&B procedure to solve the WETPSP with generalized precedence relations (GPRs) [12]. Their branching strategy resolves time infeasibility using the concepts of left shift and right shift with two fathoming rules. Ranjbar et al. [13] developed an exact method for minimizing weighted resource tardiness costs in the RCPSP. In their B&B algorithm, the branching strategy starts from a graph representing a set of conjunctions (the classical finish-start precedence constraints) and disjunctions (introduced by the resource constraints). In the B&B tree, each node is branched to two child nodes based on the two opposite directions of each undirected arc of disjunctions. Selection sequence of undirected arcs in the search tree affects the performance of the algorithm. Khoshjahan et al. [14] proposed two metaheuristics, genetic algorithm (GA), and simulated annealing (SA) to solve the RCPSP with a due date for each activity where the objective is to minimize the net present value of the earliness-tardiness penalty costs.

Kaplan [15] initially studied the preemptive RCPSP (PRCPSP) in which activities can be preempted at integer time instants and be restarted later at no additional cost. She solved PRCPSP by a reaching procedure, incorporating dominance properties and upper-lower bounds on the optimal project duration in order to decrease the amount of computational effort. Demeulemeester and Herroelen [16] developed a branch and bound algorithm for the PRCPSP. In their B&B, all activities with nonzero durations are split into subactivities, with duration of 1 and resource requirements that are equal to those of the corresponding activity. The nodes in the B&B tree correspond with partial schedules in which finish times have been assigned to a number of subactivities. Partial schedules are built up starting at time 0 and proceed systematically throughout the search process by adding at each decision point subsets of activities until a complete precedence and resource feasible schedule is obtained. Recently, Afshar-Nadjafi [17] developed a simulated annealing procedure to solve the preemptive multimode resource constrained project scheduling problem (P-MRCPSP) to minimize the project make-span subject to mode changeability after preemption.

In this work the link between PRCPSP and WETPSP is investigated. Therefore, the contribution of this paper is threefold: first, a mixed integer programming formulation is developed for the resource constrained project scheduling problem of minimizing the total earliness-tardiness penalties and preemption costs. In this problem setting, when an activity is restarted after being interrupted, a constant setup penalty is incurred with no setup time. We call this problem PRCPSP-WETPP. This model is not considered in the past literature. Second, due to NP-hardness of the problem an efficient metaheuristic solution procedure based on SA is suggested for it. In contrast to classic SA, a relatively new intelligent neighborhood search algorithm is used in this research. Finally, the efficiency of the proposed method for the PRCPSP-WETPP is evaluated statistically.

The remainder of the paper is organized as follows. Section 2 explains the resource constrained project scheduling problem of minimizing the total earliness-tardiness penalties and preemption costs (PRCPSP-WETPP). Section 3 describes the basics of simulated annealing (SA). In Section 4 the steps of proposed algorithm to solve the PRCPSP-WETPP are presented. Computational results are explained in Section 5. Finally, Section 6 concludes the paper.

2. Problem Description

The majority of previous research on project scheduling problems has investigated the problems without preemption. Although in some settings preemption is not practical, a modest degree of preemption can be accommodated in other instances. Preemption may be beneficial or harmful for the objective function. It is a well-known result in single machine scheduling theory that activity preemption may be harmful for the flow time of jobs. Whereas it is proven that preemption can reduce the make-span of a schedule in RCPSP [15], such time saving might incur a cost for making the preemptions in practice. In this paper, we assume that the preemption's penalty is constant and independent of the remaining duration of the preempted activity. Scheduling of uniprocessors can be considered as an industrial case of such a problem. Although some of these systems cannot be scheduled as a nonpreemptive scheduling problem, preemptive uniprocessor scheduling is able to successfully schedule them. In choosing between preemptive and nonpreemptive scheduling of uniprocessors, feasibility and task's due dates are essential. These schemes permit preemption where necessary for feasibility and cost reduction but attempt to avoid unnecessary preemptions during run time.

As a rule, a preemptive problem is characterized by a complicated structure of its optimal solutions. When preemption is allowed at arbitrary times, the problem turns out to be intractable [18]. Indeed, when the overall number of preemptions is unlimited, we cannot utilize exact enumerative algorithms, unless a preliminary analysis of properties of optimal solutions is performed. Such analysis reduces the original infinite set of possible points for preemption to a finite set. This analysis is performed for some scheduling problems which allow us to solve these problems by direct enumeration. Baptiste et al. [19, 20] prove that a wide class of preemptive scheduling models including both machine and project scheduling models has the "*integer preemption property*." Based on this property, for any problem instance with integral input data there exists an optimal schedule where all starting/completion times and preemptions occur at integer time points. This conclusion is held for some objective functions such as total weighted earliness-tardiness and total weighted number of late jobs.

A nonregular objective function, which is attractive in just-in-time (JIT) environments, is the minimization of the earliness-tardiness (E/T) costs of the project activities. Traditionally, tardiness penalties due to delivery after

a contractually arranged due date are considered. Tardiness leads to customer complaints, loss of reputation and profits, monetary penalties, or goodwill damages. Costs of earliness include extra storage requirements, idle times, implicitly incur opportunity costs, storage costs due to insurance, theft, perishing, and bounded capital for the case when an activity is completed before the due date. Therefore, the minimization of the earliness-tardiness (E/T) costs is attractive in order to meet the requests of practice. In this situation, each activity has a due date with associated earliness and tardiness cost. The complexity and usefulness of preemption in E/T models depend to a large extent on the type of preemption allowed and the penalty scheme applied to the different segments of each activity. In this study, based on important consequence of integer preemption property, it is assumed that accomplishing an activity can be interrupted at integer time instants and restarted at a later integer time.

In the literature of machine scheduling, different penalty schemes are studied. For example, Bülbül et al. considered only the last portion of a job may incur a penalty in a preemptive E/T scheduling problem and compared the results with other schemes [18]. We also note that such results can be used in machine environment applications, due to the nature of the manufacturing process, by simply dividing customer orders into smaller processing batches. However, they always cannot be used in project scheduling environment. In our study the penalty schemes are considered the same as the project scheduling literature [10]. Earliness and tardiness penalties for each activity are assumed to be a linear function of amount of its earliness/tardiness.

2.1. Mathematical Model. The deterministic preemptive resource constrained project scheduling problem with weighted earliness-tardiness and preemption penalties (PRCPSP-WETPP) involves the determination of start/finish times of project activities for the minimization of the total earliness-tardiness and preemption penalties of the project subject to resource and precedence constraints. Subsequently, consider a project presented in an activity on node (AON) graph $G = \{N, A\}$, where N denotes the set of nodes (activities) and A denotes the set of arcs (finish-start precedence relations with zero time lags). The activities are preemptable. Assume that the activities are topologically numbered from the start dummy activity 0 to the end dummy activity $n + 1$; that is, each predecessor of an activity has a smaller number than itself. The deterministic duration of an activity i is denoted by d_i $(1 \leq i \leq n)$, while h_i denotes its deterministic due date.

The objective of the PRCPSP-WETPP is to schedule a number of activities, for minimization of the total costs of the project considering finish to start precedence relations with a time lag of zero, constrained resources, and a predetermined deadline. We use the following symbols for PRCPSP-WETPP:

n: number of nondummy activities,

A: set of arcs of acyclic digraph representing the project,

N: set of nodes of acyclic digraph representing the project,

d_i: duration of activity i,

h_i: due date of activity i,

EST_i: earliest start time of activity i,

EFT_i: earliest finish time of activity i,

LFT_i: latest finish time of activity i,

a_k: availability of the kth resource,

r_{ik}: resource usage of activity i for resource k,

δ: deadline of the project,

Z: objective function,

π_i: each preemption penalty of activity i,

v_i: earliness cost of activity i per unit time,

τ_i: tardiness cost of activity i per unit time,

E_i: earliness of activity i (integer decision variable),

T_i: tardiness of activity i (integer decision variable).

In our formulation, 0-1 variables X_{ijt} are defined, which specify whether jth unit of duration of an activity i finishes at time t or not. Indeed, for every unit j of duration of activity i and for every feasible completion time $t \in [\mathrm{EST}_i + j, \mathrm{LFT}_i - (d_i - j)]$, X_{ijt} is defined as follows:

$$\begin{aligned} X_{ijt} &= 1, \quad \text{if } j\text{th unit of duration of activity } i \text{ finishes} \\ &\qquad \text{at time } t, \\ X_{ijt} &= 0, \quad \text{otherwise.} \end{aligned} \tag{1}$$

Also, 0-1 variables y_{ijt} are defined, which specify whether jth unit of duration of an activity i is preempted at time t or not. More specifically, for every unit j of duration of activity i and for every feasible completion time $t \in [\mathrm{EST}_i + j, \mathrm{LFT}_i - (d_i - j)]$, y_{ijt} is defined as follows:

$$\begin{aligned} y_{ijt} &= 1, \quad \text{if } j\text{th unit of duration of activity } i \text{ preempts} \\ &\qquad \text{at time } t, \\ y_{ijt} &= 0, \quad \text{otherwise.} \end{aligned} \tag{2}$$

The variables X_{ijt} and y_{ijt} can only be defined over the time interval of the activity in question. These bounds are calculated using the traditional forward and backward calculations. The backward calculation is started from a fixed project deadline δ.

Introducing the binary decision variables X_{ijt} and y_{ijt}, as well as the integer variables E_i and T_i, preemptive resource constrained project scheduling problem with weighted earliness-tardiness and preemption penalties (PRCPSP-WETPP) under the minimum total early-tardy and

preemption costs objective can be mathematically formulated as follows:

$$\min Z = \sum_{i=1}^{n} \left(v_i E_i + \tau_i T_i\right) + \sum_{i=1}^{n} \left(\pi_i \left(\sum_{j=1}^{d_i - 1} \sum_{t=\mathrm{EFT}_i - (d_i - j)}^{\mathrm{LFT}_i - (d_i - j)} y_{ijt} \right) \right), \tag{3}$$

subject to

$$E_i \geq h_i - \sum_{t=\mathrm{EFT}_i}^{\mathrm{LFT}_i} t X_{id_it}, \quad \text{for } i = 1, \ldots, n, \tag{4}$$

$$T_i \geq \sum_{t=\mathrm{EFT}_i}^{\mathrm{LFT}_i} t X_{id_it} - h_i, \quad \text{for } i = 1, \ldots, n, \tag{5}$$

$$\sum_{t=\mathrm{EFT}_i}^{\mathrm{LFT}_i} t X_{id_it} \leq \sum_{t=\mathrm{EFT}_j - (d_j - 1)}^{\mathrm{LFT}_j - (d_j - 1)} t X_{j1t} - 1, \quad \forall (i, j) \in A, \tag{6}$$

$$\sum_{t=\mathrm{EFT}_i - (d_i - j + 1)}^{\mathrm{LFT}_i - (d_i - j + 1)} t X_{i(j-1)t} + 1 \leq \sum_{t=\mathrm{EFT}_i - (d_i - j)}^{\mathrm{LFT}_i - (d_i - j)} t X_{ijt}, \quad \text{for } i = 1, \ldots, n,\ j = 2, \ldots, d_i, \tag{7}$$

$$\sum_{t=\mathrm{EST}_i + j}^{\mathrm{LFT}_i - (d_i - j)} X_{ijt} = 1, \quad \text{for } i = 1, \ldots, n,\ j = 1, \ldots, d_i, \tag{8}$$

$$X_{ijt} \leq X_{i(j+1)(t+1)} + y_{ijt}, \quad \text{for } i = 1, \ldots, n,\ j = 1, \ldots, d_i, \quad \text{for } t = \mathrm{EFT}_i - (d_i - j), \ldots, \mathrm{LFT}_i - (d_i - j), \tag{9}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{d_i - 1} r_{ik} X_{ijt} \leq a_k, \quad \text{for } k = 1, \ldots, m,\ t = 1, \ldots, \delta, \tag{10}$$

$$X_{ijt}, y_{ijt} \in \{0, 1\}, \quad \text{for } i = 1, \ldots, n,\ j = 1, \ldots, d_i, \quad \text{for } k = 1, \ldots, m,\ t = \mathrm{EST}_i + j, \ldots, \mathrm{LFT}_i - (d_i - j), \tag{11}$$

$$E_i, T_i \in \text{int}, \quad \text{for } i = 1, \ldots, n. \tag{12}$$

The objective in (3) is minimization of the total cost of the project. Equations (4) and (5) compute the earliness and tardiness of each activity. The constraint set given in (6) preserves the finish-start precedence constraints. In (7) it is specified that the finish time for each unit of duration of an activity has to be at least one time unit later than the finish time for the previous one. Equation (8) specifies that only one finish time is permitted for every unit of duration of an activity. Equation (9) guarantees that, if two successive units of duration of an activity i (i.e., units j and $j + 1$) are interrupted at time t, the corresponding decision variable y_{ijt} must set to 1. The resource constraints are specified in (10). In (11) and (12) it is specified that the decision variables X_{ijt} and y_{ijt} are binary, while E_i and T_i are integers. This formulation requires the definition of at most $2T \sum_{i=1}^{n} d_i$ binary variables and of $2n$ integer variables. Also, the number of constraints of the formulation amounts to at most $2n + n(n-1)/2 + (2 + T) \sum_{i=1}^{n} d_i + mT$.

2.2. Numerical Example. In this section, a problem instance adapted from the Patterson set [21] is used to illustrate the proposed method. The corresponding AON network is given in Figure 1. There are 7 actual and 2 dummy activities in this problem instance. One renewable resource type with availability of 5 is assumed. The numbers above the nodes denote the activities duration. Also, the three numbers below the nodes denote the due dates, the unit early-tardy costs, and the resource requirements, respectively. For the ease of representation, it is assumed that the unit earliness-tardiness costs are equal. Also, we assume the preemption penalty for all activities is equal to 1. The nonpreemptive optimal schedule for this example is shown in Figure 2 with a cost of 27. Using the LINGO version 11, based on branch and bound method, we obtained the optimal schedule of Figure 3 with a cost of 23. It is clear that one preemption for activity 1 leads to a schedule with lower cost. Although preemption has a cost of 1, it can be justified by a corresponding decrease in the tardiness penalty of activity 7 which equals 5.

2.3. NP-Hardness of the Problem. 3-PARTITION is the first problem in the theory of graphs that was proven to be NP-hard in the strong sense. In the paper by Blazewicz et al. [22], it is shown that 3-PARTITION is reducible to machine scheduling problem in which jobs with a unit processing time have to be scheduled on two parallel identical machines, where the precedence relations between the jobs are chain-like and where the jobs possibly require one unit of a single renewable resource type with an availability of one. This transformation proves that this machine scheduling problem is NP-hard in the strong sense. Also, Blazewicz et al. [22] showed that this machine scheduling problem can be transformed into an RCPSP and that any optimal solution to one problem can be transformed into an optimal solution to the other problem. As a result, NP-hardness of the RCPSP is clearly proved. However, preemptive RCPSP can be considered as an RCPSP in which each activity with duration of d_i is replaced by d_i activities with duration of 1. Therefore, NP-hardness of the PRCPSP is clearly proved.

3. Basic Simulated Annealing (SA)

Simulated annealing (SA) is a metaheuristic method that is initially presented as a search algorithm for optimization by Kirkpatrick et al. [23]. SA has been applied to several combinatorial optimization problems, including project scheduling problems [24–29]. SA starts by generating an initial solution and by setting the initial temperature parameter η. Then, at each repetition, a solution s' is created randomly in the vicinity of the current solution s. If s' overrides the current solution s, s is replaced with s'; else s is replaced with s' according to a probability computed from the Boltzmann distribution $\exp(-(f(s') - f(s))/\eta)$, where η is the current

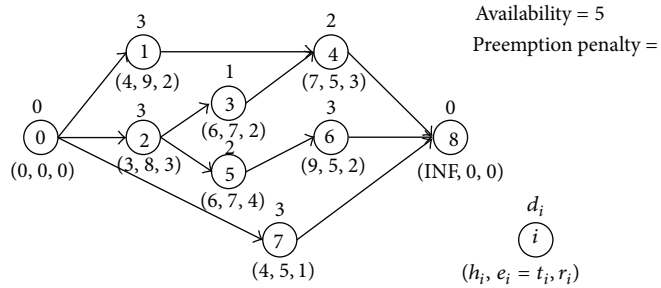

Figure 1: Problem instance for the PRCPSP-WETPP.

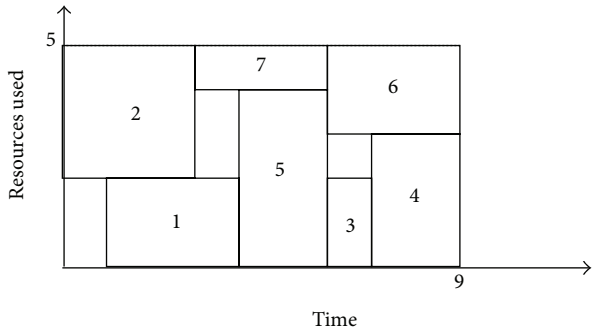

Figure 2: Optimal nonpreemptive schedule to the problem example.

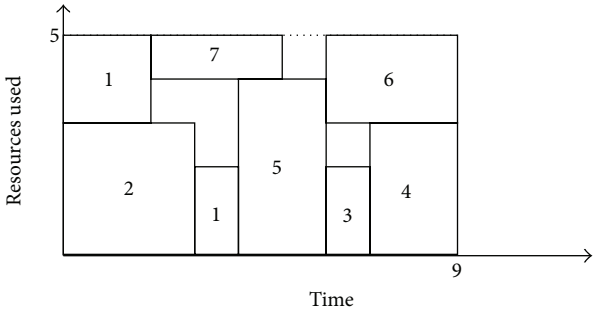

Figure 3: The optimal preemptive schedule.

temperature and $f(s') - f(s)$ is the difference between the objective function values of s and s'. It is clear that the probability of accepting uphill moves is a function of the temperature η and the difference of the objective function values $f(s') - f(s)$. At the beginning of the search process the probability of accepting worse solutions is high and it allows the exploration of the search space (random walk). During the search process this probability gradually decreases by decreasing the temperature η thus leading the search to converge to a local minimum (iterative improvement). Also, at fixed temperature, the higher the value of $f(s') - f(s)$, the lower the acceptance probability of s'. The basic structure of SA is given in Algorithm 1, where

s = the current solution,

s' = the neighbor solution,

$f(s)$ = objective value at the solution s,

η_0 = the initial temperature,

η = the current temperature.

Choosing an appropriate cooling scheme is vital for the performance of the SA. Geometric law $\eta_{k+1} = \alpha\eta_k$ is one of the most used schemes where $\alpha \in (0, 1)$. The cooling scheme can vary during the process, with the aim of balancing between diversification and intensification. For example, at the beginning of the process, η might be linearly decreasing in order to explore the search space; then, the temperature decreasing scheme may vary to the geometric in order to converge to a local minimum. The cooling scheme and the initial temperature should be tuned according to the structure of the search landscape.

The description of SA reveals that a basic SA is generally very fast. Also, it does not use the history of the search process. This is one of the reasons why SA often outperforms the other search techniques. However, due to its simplicity, SA can be integrated into other metaheuristics successfully.

4. Applying SA Algorithm to PRCPSP-WETPP

In this paper, a SA algorithm is designed to solve the above-mentioned problem. In this regard, the steps of the proposed algorithm are briefly looked into. In the sequel it is assumed that each original activity i is replaced with d_i activity with unit duration. Each duration unit $j = 1, \ldots, d_i$ of an activity i is predecessor of duration unit $(j - 1)$. Consequently, project network has $N = \sum_{i=1}^{n} d_i$ activity with duration 1 and resource requirement r_{ik} for resource type k. Only duration unit d_i^{th} has fixed due date h_i. Also, without loss of generality it is assumed that, for duration units $j = 1, \ldots, (d_i - 1)$ of an activity i, due dates are 0 with no earliness-tardiness penalty.

4.1. Solution Representation. In this paper, the activity-list (AL) is used to present a solution and a modified version of serial schedule generation scheme (SSGS) for translating a solution to a feasible schedule. The total cost criterion is a nonregular measure, that is, a measure which is not nondecreasing in activity finish time; so there is the danger of omitting optimal schedule by using the classic serial schedule generation scheme here. As a result, we apply a modified version of SSGS as follows.

The solution is represented by a vector with $N = \sum_{i=1}^{n} d_i$ elements. The elements make the AL, in which each activity

$s \leftarrow$ Generate initial solution ()
$\eta \leftarrow \eta_0$
While stopping criteria not met *do*
  $s' \leftarrow$ Pick a solution in the vicinity of s randomly
  If s' is better than s *then*
    s' replaces s
  Else
    Accept s' as new solution with probability $\exp(-(f(s') - f(s))/\eta)$
  End If
    Update (η)
End While

Algorithm 1: SA algorithm.

j has to occur before all its successors and after all its predecessors. At each stage, starting from the partial schedule assembled thus far, we select first unscheduled activity from activity list. If the selected activity is duration unit d_i^{th} of an activity i, it is inserted to partial schedule with scheduling to finish at its due date h_i or later if forced so by precedence or resource constraints; else, it is inserted to partial schedule with scheduling to finish at earliest possible time according to precedence relation and resources availabilities. This procedure continues until a complete schedule is reached.

4.2. Starting Solution. A starting solution is created randomly by setting all activities on the AL without violating the resource and precedence constraints. The starting solution is generated as follows. First, an empty $N = \sum_{i=1}^{n} d_i$ element vector is generated. To construct this vector an activity is selected randomly from set of activities that all their precedence is already met and is not selected before. This process is repeated until the AL is completed.

4.3. Neighborhood Search Procedure. An intelligent neighborhood of current solution is generated in the following procedure. A tardy list TL, is calculated which is defined as a set of activities that are scheduled to finish at later than due date forced by precedence or resource constraints. Tardy list (TL) contains the activities (with unit duration) that have tardiness with positive tardiness cost. One activity j is randomly chosen from tardy list (TL) with finish time f_j. Then, a random time instant t between $\max(\text{EFT}_j, h_j)$ and $(f_j - 1)$ is selected. Subsequently, an activity i is randomly chosen from activities with finish time $f_i = t$ such that activity i is not predecessor of activity j. Finally a pairwise interchange operator is applied which is defined as the interchange of the activities i and j by assigning new finish times $f_j' = f_i$ and $f_i' = f_j$. Activity j is scheduled to finish at time instant t and all activities between activity j and activity i are shifted to the left or right so that the precedence and resource constraints may not be violated.

4.4. Cooling Scheme. The initial temperature (η_0) of the simulated annealing algorithm is the vital factor that needs to be considered. The starting temperature is set at a large value and then reduced after each neighborhood generation, according to the cooling scheme function until it reaches the thermal equilibrium. In this paper the cooling schedule suggested by Lundy and Mees is used as follows [30]:

$$\eta_{k+1} = \frac{\eta_k}{1 + \alpha\eta_k}, \tag{13}$$

where α is selected near to zero.

4.5. Stopping Criterion. In theory the SA procedure should be continued until the final temperature η_f is zero, but in practice other stopping criteria are applied:

(i) the value of the objective function has not decreased for a large number of consecutive trials;

(ii) the number of accepted moves has become less than a certain small threshold for a large number of consecutive trials;

(iii) a fixed priori number of trials have been executed;

(iv) a maximal run time limit has been reached.

In the first two cases, the SA algorithm has a nondeterministic run time, whereas the last two cases are the cases of algorithms with a deterministic run time. In this paper, the last case is applied which is used in the literature in order to make a fair comparison between algorithms [29, 31].

5. Performance Evaluation

5.1. The Test Problems. In order to evaluate the performance of the proposed SA for the PRCPSP-WETPP, a set of 150 instances are generated by the ProGen introduced by Kolisch et al. [32]. The parameter settings of generated problems are given in Table 1.

Resource availability is assumed constant over time. For each number of activities 30 problems are generated. The resource factor (RF) reflects the average portion of resource demand per activity. The resource strength RS reflects the scarcity of the renewable resource. The unit earliness-tardiness penalty costs for each activity are generated randomly between 1 and 10. To generate the due dates of activities the method described by Vanhoucke et al. [10]

Table 1: The parameters for the generated problem set.

Control parameter	Value
Number of nondummy activities (n)	20, 30, 40, 60, and 90
Unit earliness-tardiness penalty costs	$U(1, 10)$
Preemption penalty	$U(1, 10)$
Activity durations	$U(1, 10)$
Number of start activities	3
Number of end activities	3
Maximum number of activity predecessors	3
Maximum number of activity successors	3
Coefficient of network complexity (CNC)	1.5
Resource factor (RF)	1
Resource strength (RS)	0.5
Number of renewable resource types	1, 2, and 3
Activity resource requirement (per period)	$U(1, 10)$

is used. First, a maximum due date was calculated for each project by multiplying the project critical path length by 1.5. Then, random numbers between 1 and maximum due date are generated. Finally, the generated numbers are sorted increasingly and assigned to the activities $1, 2, \ldots, n$, respectively.

5.2. Parameters Setting. The efficiency of metaheuristics depends excessively on their parameters values. In this paper, SA control parameter values are selected through the computational experiments. The CPU-time limit is applied as stopping condition. It is observed that good results are obtained by indexing the CPU-time limit to the size of the problem, that is, use of the low CPU time for small size problems and the high CPU time for large scale ones. Consequently, after some experiments, the CPU-time limit was fixed to 50 milliseconds per activity. The experiments showed that the best value for SA temperature control parameter α and initial temperature η_0 is as in Table 2.

5.3. Experimental Results. The proposed SA was coded in Borland C++ 5.02 and executed on a personal computer with an Intel Core 2 Dou, 2.5 GHz processor, and 3 GB memory. Since we could not find any algorithm for PRCPSP-WETPP, the proposed SA is compared with the optimal solution obtained by LINGO 11. The comparison results of the proposed algorithm with the optimal solution obtained by LINGO 11 (or the best obtained solution by the SA if LINGO is not able to solve the problem) are given in Table 3. Since metaheuristic algorithms are stochastic optimizers, they can provide different results for the same problem instance from one run to another. For this reason, robustness of algorithm should be analyzed based on some independent simulation runs for each problem instance. Statistically, the larger number of runs more reliably reflects the behavior of algorithm results. On the other hand, the larger number of runs increases the computational time. Frequently the used number of runs is about 5 and 10 in the literature [33, 34]. In this paper, proposed algorithm is executed 10 times for each problem to obtain more reliable data. The experimental results demonstrate that control parameter calibration provides high quality solutions.

Table 2: The tuned values for α and η_0.

Number of activities	20	30	40	60	90
α	0.0043	0.0052	0.005	0.0047	0.0047
η_0	1.2	1.15	1.35	1.35	1.4

The following definitions are used in Table 3.

NPO: number of problems for which LINGO is able to solve optimally in 1000 sec;

NPM: number of runs of problems for which SA has been able to solve optimally;

ACNT-LINGP: average convergence time for LINGO (in seconds);

ACNT-SA: average convergence time for the SA (in seconds);

ARD: average relative deviation.

Relative deviation (RD) is defined as follows:

$$\mathrm{RD} = \frac{Z - Z^*}{Z^*}, \tag{14}$$

where Z is the value of objective function obtained by SA and Z^* is the optimal objective function obtained by LINGO or the best one by the SA.

Table 3 reveals that all 60 problems with less than or equal to 30 activities have been able to solve by LINGO within the allowed CPU-time limit. Also, for problems with more than 30 activities, there are many instances that the LINGO is unable to solve, though there is a solution by the proposed SA. It can be observed that LINGO obtained optimum solutions for 79 out of 150 problems in 1000 seconds while the SA algorithm solved all 150 problems in a very shorter time and with low relative deviation. Average CPU time for LINGO indicates that by increasing the number of resource types the complexity of problem is increased. Also, ARD for the algorithm shows that proposed algorithm gives robust solutions. NPM for the proposed SA indicates that too many executions reach the optimal solution.

5.4. Managerial Insights. In this section we evaluate the effect of preemption penalty and resources availability on optimal schedule. To this end, we resolve 60 problems with 20 and 30 nondummy activities for which LINGO was able to solve optimally. We extend the parameter settings, considering two levels for resources availability (constant and variable) and three levels for preemptions cost (low, medium, and high). The previously generated preemption penalties, π_i, are applied as a medium level. The low and high levels of the preemption penalties are randomly generated between $[1, \pi_i]$ and $[\pi_i, 10]$, respectively. We assign and solve 5 problems for each combination of the parameters. The overall number of preemptions (NOPR) and the average value of objective function ($\overline{Z}$) are reported in Table 4.

Table 3: Computational results of the LINGO and SA.

Number of activities	Number of resource types	Number of problems	LINGO		SA		
			NPO	ACNT-LINGO	NPM	ARD	ACNT-SA
20	1	10	10	0.064	100	0.00%	0.108
20	2	10	10	0.132	100	0.00%	0.117
20	3	10	10	0.383	100	0.00%	0.141
30	1	10	10	3.212	100	0.00%	0.173
30	2	10	10	29.407	94	0.29%	0.211
30	3	10	10	80.545	91	0.38%	0.295
40	1	10	7	123.110	100	0.00%	0.386
40	2	10	6	200.154	92	0.46%	0.438
40	3	10	6	309.494	78	0.67%	0.516
60	1	10	0	—	92	0.59%	1.073
60	2	10	0	—	85	0.61%	1.332
60	3	10	0	—	72	0.64%	1.768
90	1	10	0	—	91	0.60%	3.650
90	2	10	0	—	86	0.79%	3.782
90	3	10	0	—	77	0.86%	4.254

Table 4: The effect of preemption cost and resource availability type.

Number of activities	Preemption cost	Resource availability			
		Constant		Variable	
		NOPR	$\overline{Z}$	NOPR	$\overline{Z}$
20	Low	87	391.65	91	375.18
20	Medium	68	427.13	74	450.62
20	High	33	476.97	35	507.12
30	Low	114	680.34	119	716.46
30	Medium	97	792.50	108	788.31
30	High	46	847.61	54	864.77

It is apparent from Table 4 that the number of preemptions is decreased when the preemption cost increases. It is because the high values of preemption penalties cannot be justified by a corresponding decrease in the earliness-tardiness penalties. However, Table 4 shows that the objective function value is an increasing function of the preemption penalty. Also, the number of preemptions in variable resource availability is slightly less than the constant case. Table 4 also reveals that the effect of resource availability type on the objective function value is not monotonously increasing or decreasing.

6. Summary and Conclusions

In this paper, the preemptive resource constrained project scheduling problem with weighted earliness-tardiness and preemption penalties (PRCPSP-WETPP) is considered. The objective of this problem is to minimize the total cost of earliness-tardiness and preemption penalties subject to the precedence relations, resource constraints, and a fixed project deadline. This problem has not been studied ever before. The problem formulated as an integer programming model, and then a simulated annealing (SA) procedure proposed to solve it. To improve the efficiency of the proposed SA, the parameters were fine-tuned. In order to generalize the statistical results, a set of 150 test problems that include problems with 20, 30, 40, 60, and 90 nondummy activities with 1, 2, and 3 resource types are generated using the ProGen software package. The efficiency of the proposed algorithm on 150 test problems was evaluated with the results of the LINGO 11. From the computation results, we could clearly see that the SA algorithm could efficiently give robust solutions for the project scheduling problem measured by the ARD. However, results displayed that the average CPU time is an increasing function of the number of activities. Also, the number of resource types has a weak negative impact on the problem complexity, measured by the required CPU time. Also, the effect of preemption penalty and resource availability type is analyzed based on 60 test problems. It reveals that increasing cost of preemptions leads to preemptions being less attractive. This research helps managers to schedule their projects in order to minimize total project costs in just-in-time (JIT) environments when preemption is permitted with penalty. Extensions of this research might be of interest, such as studying the preemptive project scheduling combined with generalized precedence relation or multimode case.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

References

[1] C. N. Potts and L. N. van Wassenhove, "Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity," *Journal of the Operational Research Society*, vol. 43, pp. 395–406, 1992.

[2] C. L. Monma and C. N. Potts, "Analysis of heuristics for preemptive parallel machine scheduling with batch setup times," *Operations Research*, vol. 41, no. 5, pp. 981–993, 1993.

[3] B. Chen, "A better heuristic for preemptive parallel machine scheduling with batch setup times," *SIAM Journal on Computing*, vol. 22, no. 6, pp. 1303–1318, 1993.

[4] S. Zdrzałka, "Preemptive scheduling with release dates, delivery times and sequence independent setup times," *European Journal of Operational Research*, vol. 76, no. 1, pp. 60–71, 1994.

[5] P. Schuurman and G. J. Woeginger, "Preemptive scheduling with job-dependent setup times," in *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 759–767, January 1999.

[6] Z. Liu and T. C. E. Cheng, "Scheduling with job release dates, delivery times and preemption penalties," *Information Processing Letters*, vol. 82, no. 2, pp. 107–111, 2002.

[7] F. M. Julien, M. J. Magazine, and N. G. Hall, "Genralized preemption models for single-machine dynamic scheduling problems," *IIE Transactions*, vol. 29, no. 5, pp. 359–372, 1997.

[8] M. Rebai, I. Kacem, and K. H. Adjallah, "Earliness-tardiness minimization on a single machine to schedule preventive maintenance tasks: metaheuristic and exact methods," *Journal of Intelligent Manufacturing*, vol. 23, no. 4, pp. 1207–1224, 2012.

[9] M. Vanhoucke, *Exact algorithms for various types of project scheduling problems non-regular objectives and time/cost trade-offs [Ph.D. dissertation]*, Katholieke Universiteit Leuven, 2001.

[10] M. Vanhoucke, E. Demeulemeester, and W. Herroelen, "An exact procedure for the resource constrained weighted earliness-tardiness project scheduling problem," *Annals of Operations Research*, vol. 102, pp. 179–196, 2000.

[11] M. Vanhoucke, E. Demeulemeester, and W. Herroelen, "Maximizing the net present value of a project with progress payments," Research Report 0028, Department of Applied Economics, Katholieke Universiteit Leuven, 2000.

[12] B. Afshar Nadjafi and S. Shadrokh, "A branch and bound algorithm for the weighted earliness-tardiness project scheduling problem with generalized precedence relations," *Scientia Iranica*, vol. 16, no. 1, pp. 55–64, 2009.

[13] 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.

[14] Y. Khoshjahan, A. A. Najafi, and B. Afshar-Nadjafi, "Resource constrained project scheduling problem with discounted earliness-tardiness penalties: mathematical modeling and solving procedure," *Computers and Industrial Engineering*, vol. 66, no. 2, pp. 293–300, 2013.

[15] L. A. Kaplan, *Resource constrained project scheduling with preemption of jobs [Ph.D. thesis]*, University of Michigan, 1988.

[16] E. L. Demeulemeester and W. S. Herroelen, "An efficient optimal solution procedure for the preemptive resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 90, no. 2, pp. 334–348, 1996.

[17] B. Afshar-Nadjafi, "A solution procedure for preemptive multi-mode project scheduling problem with mode changeability to resumption," *Applied Computing and Informatics*, 2014.

[18] K. Bülbül, P. Kaminsky, and C. Yano, "Preemption in single machine earliness/tardiness scheduling," *Journal of Scheduling*, vol. 10, no. 4-5, pp. 271–292, 2007.

[19] P. Baptiste, J. Carlier, A. Kononov, M. Queyranne, S. Sevastyanov, and M. Sviridenko, "Structural properties of optimal schedules with operation interruptions," *Diskretnyi Analizi Issledovanie Operatsii*, vol. 16, no. 1, pp. 3–36, 2009.

[20] P. Baptiste, J. Carlier, A. Kononov, M. Queyranne, S. Sevastyanov, and M. Sviridenko, "Properties of optimal schedules in preemptive shop scheduling," *Discrete Applied Mathematics*, vol. 159, no. 5, pp. 272–280, 2011.

[21] J. H. Patterson, "A comparison of exact procedures for solving the multiple constrained resource project scheduling problem," *Management Science*, vol. 30, no. 7, pp. 854–867, 1984.

[22] J. Blazewicz, 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.

[23] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[24] B. Abbasi, S. Shadrokh, and J. Arkat, "Bi-objective resource-constrained project scheduling with robustness and makespan criteria," *Applied Mathematics and Computation*, vol. 180, no. 1, pp. 146–152, 2006.

[25] K. Bouleimen and H. Lecocq, "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version," *European Journal of Operational Research*, vol. 149, no. 2, pp. 268–281, 2003.

[26] Z. He, N. Wang, T. Jia, and Y. Xu, "Simulated annealing and tabu search for multi-mode project payment scheduling," *European Journal of Operational Research*, vol. 198, no. 3, pp. 688–696, 2009.

[27] 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, pp. 137–155, 2001.

[28] M. Mika, G. Waligóra, and J. Węglarz, "Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models," *European Journal of Operational Research*, vol. 164, no. 3, pp. 639–668, 2005.

[29] A. Rahimi, H. Karimi, and B. Afshar-Nadjafi, "Using metaheuristics for project scheduling under mode identity constraints," *Applied Soft Computing*, vol. 13, no. 4, pp. 2124–2135, 2013.

[30] M. Lundy and A. Mees, "Convergence of an annealing algorithm," *Mathematical Programming*, vol. 34, no. 1, pp. 111–124, 1986.

[31] B. Naderi and H. Sadeghi, "A multi-objective simulated annealing algorithm to solving flexible no-wait flow shop scheduling problems with transportation times," *Journal of Optimization An Industrial Engineering*, vol. 5, no. 11, pp. 33–41, 2012.

[32] 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. 1693–1703, 1995.

[33] F. Forouzanfar and R. Tavakkoli-Moghaddam, "Using a genetic algorithm to optimize the total cost for a location-routing-inventory problem in a supply chain with risk pooling," *Journal of Applied Operational Research*, vol. 4, no. 1, pp. 2–13, 2012.

[34] P. Chen and S. M. Shahandashti, "Hybrid of genetic algorithm and simulated annealing for multiple project scheduling with multiple resource constraints," *Automation in Construction*, vol. 18, no. 4, pp. 434–443, 2009.