Saudi Computer Society, King Saud University

Applied Computing and Informatics

(http://computer.org.sa)
www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

A solution procedure for preemptive multi-mode project scheduling problem with mode changeability to resumption

Behrouz Afshar-Nadjafi [*]

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

Received 21 February 2013; revised 8 February 2014; accepted 25 February 2014
Available online 6 March 2014

KEYWORDS

Project scheduling;
Mode change;
Simulated Annealing;
Preemption;
Resumption

Abstract Extensive research has been devoted to the multi-mode resource constrained project scheduling problem (MRCPSP). However, little attention has been paid to problems where preemption is allowed. This paper involves the preemptive multi-mode resource constrained project scheduling problem (P-MRCPSP) to minimize the project makespan subject to mode changeability after preemption. This problem is a more realistic model and extended case of multi-mode resource constrained project scheduling problem. A binary integer programing formulation is proposed for the problem. The problem formed in this way is an NP-hard one forcing us to use the Simulated Annealing (SA) algorithm to obtain a global optimum solution or at least a satisfying one. The performance of the proposed algorithm is evaluated on 480 test problems by statistically comparing in term of the objective function and computational times. The obtained computational results indicate that the proposed algorithm is efficient and effective. Also, it is concluded from the results that mode change is very effective to improve the optimal makespan of the project.

1. Introduction

Resource constrained project scheduling problem (RCPSP) is one of the most important problems in the context of project scheduling which is an NP-hard problem [7]. The decision variables for the RCPSP are the starting times of activities while the resources availabilities are considered given. The objective is then to minimize the completion time of the project. In the literature there are several algorithms that solve the RCPSP; recent reviews about exact methods and heuristics can be found in Kolisch and Hartmann [30], Hartmann and Kolisch [19], Hartmann and Kolisch [20], Zhang et al. [48], Zhang et al. [49], Jairo et al. [23], Hartmann and Briskorn [17], Agarwal et al. [3], Fang and Wang [15], Koné [31], Paraskevopoulos et al. [38].

In RCPSP it is assumed that activities could only be performed in one possible execution mode. In practice, however, it often happens that multiple execution modes can be defined for the project activities. Each activity may be executed in one or more execution modes, each requiring a specific amount of

[*] Tel.: +98 9125817105; fax: +98 2813670051.
E-mail address: afsharnb@alum.sharif.edu.
Peer review under responsibility of King Saud University.

http://dx.doi.org/10.1016/j.aci.2014.02.003
2210-8327 © 2014 King Saud University. Production and hosting by Elsevier B.V. All rights reserved.

resources consumption and resulting in different durations for an activity completion. More exactly, each execution mode defines as a trade-off between time/cost, time/resource, speed/resource etc. The multi-mode problem (MRCPSP) is a generalized version of the RCPSP, where each activity can be performed in one out of a set of modes, with a specific activity duration and resource requirements. The standard multi-mode resource constrained project scheduling problem involves the selection of an execution mode for each activity and the determination of the activity start or finish times such that the precedence and resource constraints are met and the project duration is minimized. As this problem is a generalization of the RCPSP, the MRCPSP is also NP-hard. Several algorithms that solve the MRCPSP have been proposed in recent years: Hartmann and Drexl [18], Sprecher and Drexl [43], Knotts et al. [28], Nonobe and Ibaraki [37], Jozefowska et al. [25], Alcaraz et al. [4], Bouleimen and Lecocq [8], Heilmann [22], Zhu et al. [50], Zhang et al. [48], Zhang et al. [49], Lova et al. [32], Jarboui et al. [24], Ranjbar et al. [42], Lova et al. [33], Coelho and Vanhoucke [10], Ranjbar [41], Barrios et al. [6], Afshar-Nadjafi et al. [2], Nabipoor Afruzi et al. [36].

The basic RCPSP and MRCPSP assume that each activity, once started, will be executed until its completion. This assumption can be justified only for activities in which their interruption essentially is inapplicable. For example, in order to integrity of foundation, concrete placement cannot be preempted. However, for activities in which their interruption is applicable, the optimal makespan can be improved by allowing preemption, because the solution space is extended as a result of the constraint relaxation. Welding can be mentioned as a preemptive activity. Preemptive multi-mode resource constrained project scheduling problem (P-MRCPSP) refers to a generalization of the multi-mode resource constrained project scheduling problem (MRCPSP) which allows activities to be preempted at any time instance and restarted later on at no additional cost. The literature on solution methods for the preemptive resource constrained project scheduling problem is relatively scant. For the single-mode case, one can refer to Kaplan [26], Demeulemeester and Herroelen [13], Ballestin et al. [5], Vanhoucke and Debels [46], Damay et al. [11]. For the multi-mode case, Buddhakulsomsiri and Kim [9] proved that preemption is very effective to improve the optimal project makespan in the presence of resource vacations and temporary resource unavailability and that the makespan improvement is dependent on the parameters that impact resource utilization. Van Peteghem and Vanhoucke [47] have proposed a genetic algorithm for the multi-mode resource-constrained project scheduling problem and its extension to the preempted case.

The basic MRCPSP and P-MRCPSP assume that activities assigned modes cannot change during the execution of the project. This assumption is one of the classical MRCPSP and P-MRCPSP shortcomings. This common assumption can be justified as long as essence and materials of modes are different and mode change is inapplicable. However, if execution modes of an activity have the same essence and materials, the optimal makespan can be improved by allowing mode change. This is probable especially in the presence of resource vacations and temporary resource unavailability. However, it is likely that in reality, execution mode of an activity is changed, especially when an activity is preempted and it will be restarted at a later time. In these cases, modeling and solving such a problem as a classical MRCPSP, especially in the preemptive case may lead to poor solutions. To the best of our knowledge, no research has been performed on the P-MRCPSP with permitted mode change.

Therefore, the contribution of this paper is fourfold: first, a binary integer programing formulation is developed for the preemptive multi-mode project scheduling problem of minimizing the project makespan subject to resource constraints and precedence relations, where execution mode of each activity can be changed after being preempted. This problem is called P-MRCPSP-MC. This model is not considered in the past literature. Second, an efficient meta-heuristic solution procedure based on SA is developed for the problem due to NP-hardness of the problem. In proposed SA, the activity list representation is used to encode a project schedule and the serial schedule generation scheme (SSGS) embedded with a new dynamic heuristic to translate the schedule representation to a schedule. Third, the effectiveness of proposed SA for the P-MRCPSP-MC will be analyzed. Finally, the effect of mode changeability on project makespan is analyzed.

The remainder of this paper is organized as follows: Section 2 is devoted to the presentation of the problem. In Section 3 the steps of our algorithm to solve the problem is explained. Computational results are represented in Section 4. Finally, Section 5 contains the conclusions.

2. Problem description

In continuation the project is represented by an activity on the node (AON) network $G(N, A)$ where the set of nodes, N, represents activities and the set of arcs, A, represents finish-start precedence constraints with a time-lag of zero. The preemptable activities are numbered from the dummy start activity 1 to the dummy end activity n and are topologically ordered, i.e., each successor of an activity has a larger activity number than the activity itself. The set of activities is to be scheduled on a set R^{ρ} of renewable and R^{v} of nonrenewable resource types. For each activity $i \in N$, instead of a fixed duration and known resource requirements, a fixed work content W_i is given which essentially indicates how much work has to be performed. This work content can be performed in a mode m_i, which is chosen out of a set of M_i different execution modes, i.e., with different speeds and resource requirements as long as the required work content is met. The accomplishing of an activity can be temporarily interrupted at discrete time instants, and restarted at a later time with a same or different mode. The progress of activity i during each time unit of its execution in mode m_i , is w_{im_i} (measured by same unit of W_i). Each activity i in mode m_i requires $r^{\rho}_{im_ik}$ renewable resource units ($k \in R^{\rho}$) during each time unit of its execution. For each renewable resource$k \in R^{\rho}$, the availability a^{ρ}_{k} is constant throughout the project horizon. Activity i, executed in mode m_i, will also use $r^{v}_{im_il}$ nonrenewable resource units ($l \in R^{v}$) of the total available nonrenewable resource a^{v}_{l}. Logically, it is assumed that mode m_i with higher w_{im_i} requires more renewable $r^{\rho}_{im_ik}$ and nonrenewable resources $r^{v}_{im_il}$.

The objective of the P-MRCPSP-MC is to find a feasible schedule in order to minimize the makespan of the project. However, changeable execution modes m_i for activities and preemption plan for activities have to be determined. A schedule is defined as a sequence of start (finish) times for the project's activities. A schedule which satisfies the specified precedence and resource constraints is called feasible schedule. Also, a feasible schedule which meets as much as possible the objectives set forward by project management is called optimal. The following notation is used for P-MRCPSP-MC:

A	Set of arcs of acyclic digraph representing the project
N	Set of nodes of acyclic digraph representing the project, $\|N\| = n$
n	Number of activities, index by i
R^ρ	Set of renewable resource(s), $\|R^\rho\| = K$
K	Number of renewable resource(s), index by k
R^v	Set of nonrenewable resource(s), $\|R^v\| = L$
L	Number of nonrenewable resource(s), index by l
M_i	Number of execution modes for activity i , index by m_i
W_i	Total work content of activity i, $i \in N$
w_{im_i}	Progress of activity i during each time unit of its execution in mode m_i , $i \in N$, $m_i = 1,\ldots,M_i$
$r^\rho_{im_ik}$	Renewable resource type k requirement of activity i in mode m_i , $i \in N$, $k \in R^\rho$, $m_i = 1,\ldots,M_i$
$r^v_{im_il}$	Nonrenewable resource type 1 requirement of activity i in mode m_i, $i \in N$, $l \in R^v$, $m_i = 1,\ldots,M_i$
a^ρ_k	Constant availability of renewable resource type k throughout the project horizon, $k \in R^\rho$
a^v_l	Total availability of nonrenewable resource type l, $l \in R^v$
EST_i	Earliest start time of activity i
LFT_i	Latest finish time of activity i
τ_n	Deadline of the project
Z	Objective function (project makespan)
x_{im_it}	1, if activity i in mode m_i is in progress in period t, 0, otherwise (binary decision variable)

In our formulation, 0–1 variables x_{im_it} are defined, which specify whether an activity i in mode m_i is in progress in period t or not. These variables can only be defined over the time interval of the activity in question $t \in [EST_i + 1, LFT_i]$. These limits are determined using the traditional forward and backward pass calculations considering duration of activity i based on high speed mode as follows:

$$d_i = \left\lceil \frac{W_i}{\max\limits_{m_i} w_{im_i}} \right\rceil \tag{1}$$

The backward pass calculation is started from a fixed project deadline τ_n. In this paper, earliest finish time of dummy end activity, EFT_n, is considered as project deadline. EFT_n is computed using the traditional forward calculations considering duration of activity i based on low speed mode as follows:

$$d_i = \left\lceil \frac{W_i}{\min\limits_{m_i} w_{im_i}} \right\rceil \tag{2}$$

It is clear that an activity with work content of 0 is never in progress and thus does not have a corresponding decision variable which is set to 1. This problem, however, can be easily overcome: the dummy start and end activity are assigned a dummy mode with work content of 1. Also, the parameters $r^v_{im_il}$, $r^\rho_{im_ik}$ and w_{im_i} for dummy modes are assumed as 1. All other activities with zero work content can be eliminated, provided that the corresponding precedence relations are adjusted appropriately. The resulting schedule may be transferred into a schedule for the original problem by removing the dummy start and end activity, and one time unit left shifting.

Using the above notation, P-MRCPSP-MC can be mathematically formulated as follows:

$$\min Z = \sum_{t=EST_n+1}^{LFT_n} t.x_{nm_nt} \tag{3}$$

Subject to:

$$\sum_{t=EST_i+1}^{LFT_i} \sum_{m_i=1}^{M_i} w_{im_i}.x_{im_it} \geq W_i \quad \forall i \in N \tag{4}$$

$$\sum_{m_i=1}^{M_i} x_{im_it} \leq 1 \quad \forall i \in N, \forall t \in [EST_i + 1, LFT_i] \tag{5}$$

$$x_{im_it} + x_{im'_i(t+1)} \leq 1 \quad \forall i \in N, \quad \forall t \in [EST_i + 1, LFT_i - 1], \forall m_i \neq m'_i \tag{6}$$

$$\sum_{\tau=1}^{t} \sum_{m_i=1}^{M_i} w_{im_i}.x_{im_i\tau} \geq W_i \sum_{m_j=1}^{M_j} x_{jm_j(t+1)} \quad \forall (i,j) \in A, \quad \forall t \in [EST_i + 1, LFT_j - 1] \tag{7}$$

$$\sum_{i=1}^{n} \sum_{m_i=1}^{M_i} r^\rho_{im_ik}.x_{im_it} \leq a^\rho_k \quad \forall k \in R^\rho, t = 1, \ldots, \tau_n \tag{8}$$

$$\sum_{i=1}^{n} \sum_{m_i=1}^{M_i} \sum_{t=EST_i+1}^{EFT_i} r^v_{im_il}.x_{im_it} \leq a^v_l \quad \forall l \in R^v \tag{9}$$

$$x_{im_it} = 0, 1 \quad \forall i \in N, \forall t \in [EST_i + 1, LFT_i], \quad m_i = 1, \ldots, M_i \tag{10}$$

The objective function in Eq. (3) minimizes the project duration. Remember, however, that this value exceeds the optimal project length because of the unit duration of both the dummy start and dummy end activity. The constraints in inequality (4) assure that work content of each activity is met. The constraints in Eq. (5) assure that each activity is not assigned more than one mode for each time period. The assumption that mode change is not allowed without preemption is modeled in Eq. (6). Eq. (7) denotes the precedence relations-constraints. Constraints (8) and (9) take care of the renewable and nonrenewable resource limitations, respectively. Finally, Eq. (10) imposes binary values on the decision variables. This formulation requires the definition of at most n^* $\max(M_i)^*$ τ_n binary decision variables. Also, the number of constraints of the formulation amounts to at most $n + n\ \tau_n$ $[1 + \max(M_i)(\max(M_i)\text{-}1)/2] +$ τ_n $(|A| + K) + L$.

Fig. 1 shows an example of P-MRCPSP-MC with 7 activities where 1 and 7 are dummy activities.

Each activity has two execution modes. For each mode, 1 renewable resource and 1 nonrenewable resource is indicated. The availability for the renewable (nonrenewable) resource is 5 (132). Problem instance parameters are given in Table 1.

Fig. 2(a) depicts a schedule with a makespan of 8 days. This schedule is feasible because it uses exactly 130 nonrenewable resource units. Also, precedence relations are met and renewable resource availability (5) is not violated. Fig. 2(b) shows

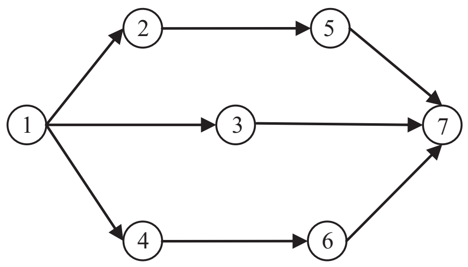

Fig. 1 An example network.

Table 1 Problem instance information.

Activity i	W_i	Mode m_i	w_{im_i}	$r^{\rho}_{im_ik}$	$r^{v}_{im_il}$
1	1	1	1	0	0
2	14	1	6	5	10
		2	4	4	7
3	8	1	5	2	13
		2	3	1	6
4	11	1	8	4	23
		2	5	2	10
5	14	1	7	3	7
		2	5	2	5
6	6	1	6	2	22
		2	3	1	11
7	1	1	1	0	0

a feasible schedule with a makespan of 7 days, in which preemption is allowed. If the problem is relaxed to the P-MRCPSP-MC, a feasible schedule as shown in Fig. 2(c) can be generated.

3. Proposed SA to solve P-MRCPSP-MC

Simulated Annealing (SA) algorithm has been successfully applied to a noticeable number of project scheduling problems [1,8,21,25,35,40]. In this section an SA algorithm is proposed to solve P-MRCPSP-MC. In order to increase quality of the proposed SA, an efficient dynamic heuristic algorithm is implemented to construct a schedule. Also exact solutions obtained from Lingo 11 are considered to provide comparable computational efforts for SA.

3.1. Basic Simulated Annealing

Simulated Annealing (SA) which has been successfully applied to various difficult combinatorial optimization problems is a random search method that is based on Monte Carlo iterative strategy. The origins of SA are in statistical mechanics (Metropolis algorithm) and it initially was presented as a search algorithm for combinatorial optimization by Kirkpatrick et al. [27]. SA is useful for problems with a very large discrete search space, which is too large for an enumeration search method. SA algorithm starts by generating an initial solution and by initializing the so-called temperature parameter T. Then, at each iteration a solution s' is randomly created in the neighborhood of the current solution and if it is better than the current solution, it replaces the current solution. If the new solution is not an improvement upon the current solution, it replaces the current solution with a probability generally computed following the Boltzmann distribution $\exp(-\frac{f(s')-f(s)}{T})$ where T is the current temperature and $f(s')$ - $f(s)$ is the change in objective function value obtained by moving from previous solution to new solution. The temperature T is decreased during the search process, thus at the beginning of the search the probability of accepting uphill moves is high and it gradually decreases, converging to a simple iterative improvement algorithm. Regarding the search process, this means that the algorithm is the result of two combined strategies: random walk and iterative improvement. In the first phase of the search, the bias toward improvements is low and it permits the exploration of the search space; this erratic component is slowly decreased thus

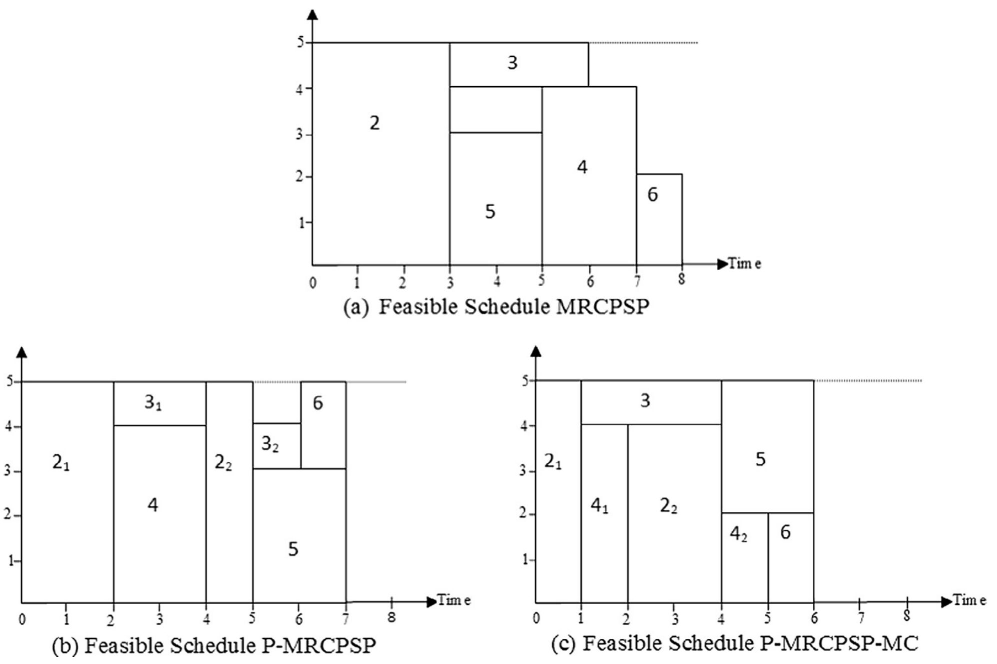

Fig. 2 Schedules for problem instance.

leading the search to converge to a (local) minimum. The probability of accepting uphill moves is controlled by two factors: the difference of the objective functions $f(s')$ - $f(s)$ and the temperature T. On the one hand, at fixed temperature, the higher the difference $f(s')$ - $f(s)$, the lower the probability to accept a move from s to s'. Whereas, the higher T, the higher the probability of uphill moves.

The choice of an appropriate cooling schedule is crucial for the performance of the algorithm. One of the most used ones follows a geometric law $T_{k+1} = \alpha T_k$ where $\alpha \in (0, 1)$ which corresponds to an exponential decay of the temperature. The cooling rule may vary during the search, with the aim of tuning the balance between diversification and intensification. For example, at the beginning of the search, T might be constant or linearly decreasing, in order to sample the search space; then, T might follow a rule such as the geometric one, to converge to a local minimum at the end of the search. The cooling schedule and the initial temperature should be adapted to the particular problem instance, since the cost of escaping from local minima depends on the structure of the search landscape. The description of SA indicates that a basic SA does not use the history of the search process. This is one of the reasons why SA is often outperformed by other meta-heuristics. However, due to its simplicity, it is generally very fast and it can be successfully integrated into other search techniques.

3.2. Preprocessing

In order to reduce the search space, preprocessing is used before the execution of SA. The data reduction procedure has originally been proposed by Sprecher et al. [44] to increase the speed of their branch and bound algorithm for the MRCPSP. The idea behind this procedure is to omit all non-executable and inefficient modes from the project data without affecting the optimal makespan. An execution mode m_j is called *non-executable* if its execution would violate the renewable resource constraints in any schedule. Also, a mode is called *inefficient* if there is another mode of the same activity with the same or higher speed and no more requirements for all resources. Hence, non-executable and inefficient modes may be excluded from the project data without losing optimality.

3.3. Solution representation

In the previous researches, various representations for schedules in the construction of heuristics for the RCPSP are developed (Kolisch and Hartmann [30]). The two most important ones are the random-key (RK) representation and the activity-list (AL) representation. Hartmann and Kolisch [19] deduced from experimental tests that procedures based on AL representations outperform the other procedures. The AL representation is used to encode a project schedule and the serial schedule generation scheme (SSGS) to translate the schedule representation to a schedule. Since the minimum project makespan criterion is a regular performance measure, i.e., a measure which is non-decreasing in activity completion times, one may use the serial SGS rule to construct the schedule. As a result, there is no danger of omitting an optimal schedule by using the serial SGS here. The serial SGS sequentially adds activities to the schedule until a feasible complete schedule is obtained. In each run, the first un-scheduled activity in the activity list is chosen and the first possible starting time is assigned for that activity such that precedence or resource constraints are preserved.

A feasible solution is represented by a vector which is a precedence-feasible permutation of activities:

$$I = (j_i^I, j_2^I, \ldots, j_n^I) \tag{11}$$

Fig. 3 shows an example of solution representation related to the mentioned instance in Fig. 1.

Having got a feasible solution represented by the vector described above, the starting times of all activities (sub-activities) are then defined by using the serial SGS. The SGS determines how a feasible schedule is constructed by assigning starting times to the activities. It sequentially adds activities in the activity list to the schedule until a feasible complete schedule is obtained such that no precedence or resource constraint is violated. In this paper however, execution modes are determined using a dynamic heuristic embedded into serial SGS. Our proposed heuristic is derived from part period lot sizing heuristic (DeMatteis [12]); algorithm chooses the number of periods covered by the replenishment order such that the total holding costs are made as close as possible to the setup cost.

To solve P-MRCPSP-MC, an initial feasible activity list is generated. Then, ratio of work to resource (RWR) is computed as follows:

$$RWR = \frac{\sum_{i=1}^{n} W_i}{\sum_{l=1}^{L} a_l^v + T\sum_{k=1}^{K} a_k^\rho} \tag{12}$$

RWR is a representation of work-resource balancing. In our proposed SA, mode assignment is done by comparing realized ratio of work to resources so far with RWR. For selected activity i, mode m_i with realized ratio of total completed work content to the spent resources so far as close as to RWR is assigned.

After obtaining an initial activity list I, the corresponding schedule is computed by the following procedure:

Starting from time period 1, for each time period t, set of activities that are executable at a certain time period t (i.e., all their predecessors have completed), is identified. This set of activities is denoted by I_a which should consist of at least one activity. Activities in I_a are arranged according their sequence in I. In each iteration, first activity j is selected from I_a and deleted from it. If activity j was in progress at period t-1, same execution mode should be assigned to it if possible. Else, procedure is continued by selecting first activity j from I_a again.

If activity j was in progress at period t-1, all *impossible* and *dominated* modes of activity j are considered inactive. A mode m_j is called *impossible* if its renewable resource requirement exceeds remaining availability for at least one renewable resource type. Also, A mode m_j is *dominated* by another mode m_j' if remaining work of activity j is less than or equal to $w_{jm_j'}$ whereas $w_{jm_j} > w_{jm_j'}$. Then, for remaining modes (if exist) of

Activity list

j_1	j_2	j_3	j_4	j_5	j_6	j_7
1	2	4	3	5	6	7

Fig. 3 Solution representation.

activity j, P_{jm_j} is computed as follows which is a representation of realized ratio of work to resources, so far.

$$P_{jm_j} = \frac{\sum_i \sum_{m_i} \sum_{\tau=1}^{t} w_{im_i} . x_{im_i\tau}}{\sum_i \sum_{m_i} \sum_l \sum_{\tau=1}^{t} r^{v}_{im_il} x_{im_i\tau} + t\sum_{k=1}^{K} a^{\rho}_{k}} \tag{13}$$

Finally, the mode m_j for which P_{jm_j} most nearly equals to RWR is assigned to activity j and activity j set to be in progress at period t . If there are no remaining modes for activity j, procedure continues by selecting first activity j is from I_a. Also, if I_a is empty, algorithm restarts by setting $t = t + 1$. Above procedure is continued until dummy activity n be a scheduled activity. The time complexity of this procedure is the same as the basic serial SGS, $O(n^2K)$ (Pinson et al. [39]). The pseudo-code for decoding a solution to a schedule is shown in Table 2. For example, in Fig. 1 consider the following activity list:

j_1	j_2	j_3	j_4	j_5	j_6	j_7
1	2	4	3	5	6	7

Decoding this schedule as follows results in the feasible schedule shown in Fig. 2(c):

$t = 1$ activity 2 in mode m_1
$t = 2$ activity 4 in mode m_1
$t = 2$ activity 3 in mode m_2
$t = 3$ activity 2 in mode m_2
(activity 2 is preempted at time point $t = 1$)
$t = 3$ activity 3 in mode m_2
$t = 4$ activity 2 in mode m_2
$t = 4$ activity 3 in mode m_2
$t = 5$ activity 4 in mode m_1
(activity 4 is preempted at time point $t = 2$)
$t = 5$ activity 5 in mode m_1
$t = 6$ activity 5 in mode m_1
$t = 6$ activity 6 in mode m_1

The number of requested nonrenewable resource units that exceeds the capacity $a^{v}_{l}, l \in R^{\tau}$, is defined as the excess of resource request $ERR(\mu)$ (Van Peteghem and Vanhoucke [47]). An $ERR(\mu) = 0$ means that the solution is feasible. If $ERR(\mu)$ is larger than 0, the solution is infeasible with respect to nonrenewable resources. The formula of the $ERR(\mu)$ can be adjusted in our problem as follows:

$$ERR(\mu) = \sum_{l=1}^{L} \left(\max \left(0, \sum_{j=1}^{n} \sum_{t=1}^{\tau_n} \sum_{m_j=1}^{M_j} (r^{v}_{jm_jl} . x_{jm_jt}) - a^{v}_{l} \right) \right) \tag{14}$$

For an infeasible generated schedule ($ERR(\mu) > 0$) the local search procedure of Hartmann [16] is applied to transform infeasible solutions into feasible ones. The procedure chooses an activity randomly and for that activity, a different mode is chosen. If the $ERR(\mu)$ remains the same or decreases, the mode for that activity is changed. This step is repeated until the mode assignment is feasible ($ERR(\mu) = 0$) or until J consecutive unsuccessful trials to improve the mode assignment have been made. In this paper, J equals to four times the number of activities in the project. This procedure acts only on mode assignment and do not change the activity list. Also, this procedure stops as soon as it reaches to a feasible solution; i.e., resulting solution is close to the inner border of nonrenewable feasibility. So this procedure may not generate a feasible solution that is very different from the original unfeasible one.

3.4. Starting solution

In the proposed SA an initial solution is created by setting all activities on the activity list based on the latest finish time (LFT) which is an efficient priority rule (Kolisch [29]). Then, the procedure described in Section 4.2 is used to determine execution modes and execution time of activities.

3.5. Neighborhood generation structure

In order to generate a neighborhood of current solution the following method is used. Let $I = (j^I_i, j^I_2, \ldots, j^I_n)$ be the current solution. Neighborhood generation mechanism is applied to the activity list of the solution. For activity list of I, neighborhood generation mechanism operates as follows: A random activity j^I_a, is selected from the activity list with position a. Last predecessor and first successor's position of j^I_a is identified in the activity list. Subsequently, a random position x between the last predecessor and first successor's position of j^I_a is selected, and j^I_a is moved to position x. Finally, all activities between position of j^I_a and position x are shifted to the left or right depending on relative position of j^I_a and position x.

3.6. Cooling scheme

The cooling scheme is the main factor that needs to be organized when designing the Simulated Annealing algorithm. The temperature is initially set at a large value and then gradually decreased under the cooling schedule function until it reaches the thermal equilibrium. After each move (neighborhood generation), the temperature is reduced according to

Table 2 Pseudo-code for decoding a solution to a schedule.

```
1) Let $t = 1$, compute $RWR$ from Eq. (12)
2) Determine the list of uncompleted activities which are admissible to be in progress at period $t$ with respect to the precedence constraints, ($I_a$). Arrange activities in $I_a$ according their sequence in $I$
3) If $I_a$ is empty set $t = t + 1$ and go to step 2, else, select first activity $j$ from $I_a$ , Delete $j$ from $I_a$ , If $j = n$ go to step 8
4) If activity $j$ was in progress at period $t$-1, assign the same execution mode to it if possible and go step 7 else, go to step 3
5) Delete all impossible and dominated modes of activity $j$. If the remaining mode list of activity $j$ is empty go to step 3
6) For remaining modes of activity $j$ compute $P_{jm_j}$ from Eq. (13), assign the mode $m_j$ to activity for which $P_{jm_j}$ is most nearly equal to $RWR$
7) Set activity $j$ in assigned mode to be in progress at period $t$, go to step 2
8) Stop
```

the cooling schedule suggested by Lundy and Mees [34] as follows where α is chosen close to zero.

$$T_{k+1} = \frac{T_k}{1 + \alpha T_k} \tag{15}$$

3.7. Stopping criterion

In theory the SA procedure should be continued until the final temperature T_f is zero, but in practice other stopping criteria are used. In this paper, the procedure is continued until a predetermined CPU time is reached.

4. Performance evaluation

4.1. The test problems

A set of 480 problems was generated by the project generator ProGen/πx developed by Drexl et al. [14] in order to validate the proposed SA algorithm for the P-MRCPSP-MC. To do this the parameters given in Table 3 are used. The indication [*x*,*y*] means that the value is randomly generated on the interval [*x*,*y*]. Renewable resource availability is constant over time. For each combination of the parameter values, 4 instances were generated. The resource factor *RF* reflects the average portion of resource required per activity. The resource strength *RS* reflects the scarceness of the resource. The problem set was extended by generating project deadline τ_n in the same way as described in Section 2.

4.2. Parameters setting

The values of parameters used in Simulated Annealing (SA) algorithms must be carefully selected since parameter values may have a significant influence on the performance of the algorithm. In this paper, the Taguchi experimental design is used to tune the parameters of SA. CPU-time limit was specified as a stopping criterion which is selected through the computational experiments. We obtained good results by indexing the CPU-time limit to the size of the problem, i.e., use of the low CPU-time for small problems and high CPU-time for larger problems. Taguchi [45] divides factors into controllable and noise factors and offer a set of orthogonal arrays for designing experiments of quality improvement. Although there is no direct control of noise factors, the Taguchi method determines the optimal level of controllable factors and minimizes the effect of noise. In the proposed SA, the factors that should be tuned are temperature control parameter α, initial temperature and number of milliseconds per activity *CPU*. A set of 27 randomly generated problems with 40 non-dummy activities are used for parameter tuning. Using MINITAB software version 16, based on a L27 orthogonal array design the optimal levels (in **Bold**) of the parameters are reported in Table 4.

4.3. Experimental results

The procedure has been programed in Borland C+ + 5.02 and executed on a personal computer with an Intel Core2Dou, 2.5 GHz processor and 3 GB memory. Since we could not find any algorithm for P-MRCPSP-MC, the proposed SA is compared with the optimal solution obtained by Lingo 11. Table 5 presents the computational results of the proposed algorithm where it is compared with the optimal solution obtained by Lingo 11 (or the best obtained solution by SA if Lingo is not able to solve the problem). Proposed SA executed 10 times for each problem to obtain more reliable data. The experimental results demonstrate that control parameter calibration provides high quality solutions. Following notations are used in Table 5:

NPO: Number of problems for which Lingo was able to find optimum solution in 1000 s.
NPM: Number of runs of problems for which SA was able to find optimum solution.
ACNT-L: Average convergence time for Lingo (in seconds).
ACNT-SA: Average convergence time for SA (in seconds).
ARD: Average relative deviation percentages.

Relative deviation (*RD*) percentage for each problem is obtained by following formula:

$$RD = \frac{Z - Z^*}{Z^*} \tag{16}$$

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

From Table 5 it can be observed that when the number of activities is less than or equal to 30, all 192 problems can be solved to optimality by Lingo within the allowed time limit. Also, Table 5 shows that when number of activities is greater

Table 3 The parameter settings for the problem set.

Control parameter	Value
Number of activities (non-dummy) (*n*)	20, 30, 40, 60, 90
Number of execution modes	2, 3
Activity work contents	[10,20]
Progress of activities (per period)	[1,10]
Number of initial and terminal activities	3
Maximal number of predecessors and successors	3
Coefficient of network complexity (*CNC*)	1.5
Resource factor (*RF*)	0.5, 1
Renewable and nonrenewable resource strength (*RS*)	0.25, 0.5
Number of renewable and nonrenewable resource types	1, 2, 3
Constant availability of renewable resources	*n*
Total availability of nonrenewable resource	25*n*
Activity renewable resource (per period) demand	Integer[1,10]
Activity nonrenewable resource (per period) demand	Integer[1,10]

Table 4 Factors levels and the tuned values for α, T_0 and *CPU*.

Factor	Number of Levels	Level 1	Level 2	Level 3
CPU	3	50	**70**	100
α	3	0.0045	**0.0055**	0.0065
T_0	3	12	**17**	25

Table 5 Computational results of the SA and Lingo.

#Activities	#Modes	#Problems	SA			Lingo	
			NPM	ARD (%)	ACNT-SA	NPO	ACNT-L
20	2	48	480	0.00	0.014	48	3.96
20	3	48	480	0.00	0.023	48	5.62
30	2	48	480	0.00	0.079	48	37.19
30	3	48	392	0.34	0.053	48	68.27
40	2	48	480	0.00	0.069	16	107.30
40	3	48	381	0.39	0.074	10	132.61
60	2	48	354	0.42	0.189	9	265.78
60	3	48	335	0.56	0.274	2	324.43
90	2	48	351	0.50	0.190	0	–
90	3	48	321	0.55	0.329	0	–

Table 6 Computational results of the Lingo with and without mode change.

#Activities	#Modes	#Problems	P-MRCPSP-MC vs. P-MRCPSP		
			Average improvement (%)	Better	Equal
20	2	48	4.32	46	2
20	3	48	5.76	48	0
30	2	48	5.11	45	3
30	3	48	6.85	47	1
40	2	16	7.64	15	1
40	3	10	9.46	10	0
60	2	9	8.39	9	0
60	3	2	9.82	2	0

than 30, while there are many instances that the Lingo is unable to solve, there is a solution by SA. However, Lingo obtained optimum solutions for 229 out of 480 problems in 1000 s and SA algorithm solved all problems with low relative deviation and in a very short time (70 ms per activity). Average CPU-time for Lingo indicates that when the number of execution modes is increased the complexity of the problem is increased. ARD for the algorithm shows that proposed SA gives robust solutions. Also, NPM for the algorithm indicates that too many executions of problems reach the optimum solution.

To observe the statistical comparison of the differences between the results of the Lingo and SA, a paired *t*-test is used for 229 problems to which Lingo obtained optimum solutions and the corresponding 95% confidence interval is calculated as [−9.36, 7.82]. Since the lower confidence level is negative and the upper level is positive, then the null hypothesis cannot be rejected as the population mean of the differences could be zero. This implies that the differences between the quality of solutions obtained by Lingo and SA are not statistically significant.

In order to evaluate the effect of changeability assumption, for 229 problems which Lingo obtained optimum solutions, each problem has been solved without mode changeability assumption. Table 6 presents the computational results. From Table 6 it can be observed that mode changeability obviously leads to an overall average makespan improvement. Table 6 also reveals that mode changeability usually leads to better solutions. Average Improvement (%) column shows that the percent loss due to using the P-MRCPSP model instead of the P-MRCPSP-MC is straightly relevant to number of activities and execution modes.

5. Summary and conclusions

The preemptive multi-mode resource constrained project scheduling problem with permitted mode change (P-MPRCPSP-MC), is investigated in this paper. The objective of P-MPRCPSP-MC is to schedule the activities in order to minimize the project makespan subject to the precedence constraints and resource constraints. In this problem setting, work content concept is used instead of duration. This problem has not been studied ever before. The problem described with an integer programing model, and then the parameters tuned Simulated Annealing (SA) proposed to solve it. The performance of the proposed algorithm on 480 test problems was compared with the results of the Lingo 11. From the computation results, one could clearly see that the SA algorithm could efficiently solve the project scheduling problem. Also, one could find out that mode changeability obviously leads to an average makespan improvement. For further research, we recommend the adapting mode change concept for other extensions of multi-mode project scheduling problems.

References

[1] B. Abbasi, S. Shadrokh, J. Arkat, Bi-objective resource constrained project scheduling with robustness and makespan criteria, Appl. Math. Comput. 180 (2006) 146–152.

[2] B. Afshar-Nadjafi, A. Rahimi, H. Karimi, A genetic algorithm for mode identity and the resource constrained project scheduling problem, Sci. Iran. 20 (3) (2013) 824–831.

[3] A. Agarwal, S. Colak, S. Erenguc, A neurogenetic approach for the resource constrained project scheduling problem, Comput. Oper. Res. 38 (1) (2011) 44–50.

[4] J. Alcaraz, C. Maroto, R. Ruiz, Solving the multi-mode resource constrained project scheduling problem with genetic algorithms, J. Oper. Res. Soc. 54 (2003) 614–626.

[5] F. Ballestin, V. Valls, S. Quintanilla, Pre-emption in resource constrained project scheduling, Eur. J. Oper. Res. 189 (2008) 1136–1152.

[6] A. Barrios, F. Ballestin, V. Valls, A double genetic algorithm for the MRCPSP/max, Comput. Oper. Res. 38 (1) (2011) 33–43.

[7] J. Blazewicz, J. Lenstra, A. Rinnooy Kan, Scheduling subject to resource constraints: classification and complexity, Discrete. Appl. Math. 5 (1983) 11–24.

[8] K. Bouleimen, H. Lecocq, A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, Eur. J. Oper. Res. 149 (2003) 268–281.

[9] J. Buddhakulsomsiri, D. Kim, Properties of multi-mode resource constrained project scheduling problems with resource vacations and activity splitting, Eur. J. Oper. Res. 175 (2006) 279–295.

[10] J. Coelho, M. Vanhoucke, Multi-mode resource constrained project scheduling using RCPSP and SAT solvers, Eur. J. Oper. Res. 213 (1) (2011) 73–82.

[11] J. Damay, A. Quilliot, E. Sanlaville, Linear programming based algorithms for preemptive and non-preemptive RCPSP, Eur. J. Oper. Res. 182 (2007) 1012–1022.

[12] J.J. DeMatteis, The part-period algorithm, IBM Syst. J. 7 (1) (1968) 30–38.

[13] E. Demeulemeester, W. Herroelen, An efficient optimal procedure for the preemptive resource-constrained project scheduling problem, Eur. J. Oper. Res. 90 (1996) 334–348.

[14] A. Drexl, R. Nissen, J.H. Patterson, F. Salewski, ProGen/πx – An instance generator for resource constrained project scheduling problems with partially renewable resources and further extensions, Eur. J. Oper. Res. 125 (2000) 59–72.

[15] C. Fang, L. Wang, An effective shuffled frog-leaping algorithm for resource constrained project scheduling problem, Comput. Oper. Res. 39 (5) (2012) 890–901.

[16] S. Hartmann, Project scheduling with multiple modes: a genetic algorithm, Ann. Oper. Res. 102 (2001) 111–135.

[17] S. Hartmann, D.A. Briskorn, Survey of variants and extensions of the resource constrained project scheduling problem, Eur. J. Oper. Res. 207 (1) (2010) 1–14.

[18] S. Hartmann, A. Drexl, Project scheduling with multiple modes: a comparison of exact algorithms, Networks 32 (1998) 283–297.

[19] S. Hartmann, R. Kolisch, Experimental evaluation of state-of-the-art heuristics for the resource constrained project scheduling problem, Eur. J. Oper. Res. 127 (2000) 394–407.

[20] S. Hartmann, R. Kolisch, Experimental investigation of heuristics for resource-constrained project scheduling: an update, Eur. J. Oper. Res. 17 (2006) 23–37.

[21] Z. He, N. Wang, T. Jia, Y. Xu, Simulated annealing and tabu search for multimode project payment scheduling, Eur. J. Oper. Res. 198 (2009) 688–696.

[22] R. Heilmann, A branch-and-bound procedure for the multi-mode resource constrained project scheduling problem with minimum and maximum time lags, Eur. J. Oper. Res. 144 (2) (2003) 348–365.

[23] R. Jairo, M. Torres, G.F. Edgar, C. Pirachicán-Mayorga, Project scheduling with limited resources using a genetic algorithm, Int. J. Project Manage. 28 (6) (2010) 619–628.

[24] B. Jarboui, N. Damak, P. Siarry, A. Rebai, A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems, Appl. Math. Comput. 195 (2008) 299–308.

[25] J. Jozefowska, M. Mika, R. Rozycki, G. Waligora, J. Weglarz, Simulated annealing for multi-mode resource constrained project scheduling, Ann. Oper. Res. 102 (2001) 137–155.

[26] L. Kaplan, Resource-Constrained Project Scheduling with Preemption of Jobs (Ph.D. Thesis), University of Michigan, 1988.

[27] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.

[28] G. Knotts, M. Dror, B. Hartman, Agent-based project scheduling, IIE Trans. 32 (5) (2000) 387–401.

[29] R. Kolisch, Serial and parallel resource constrained project scheduling methods revisited: theory and computation, Eur. J. Oper. Res. 90 (1996) 320–333.

[30] R. Kolisch, S. Hartmann, Heuristic algorithms for solving the resource constrained project scheduling problem: classification and computational analysis, in: J. Weglarz (Ed.), Project Scheduling: Recent Models, Algorithms and Applications, Kluwer Academic Publishers, Boston, 1999, pp. 147–178.

[31] O. Koné, New approaches for solving the resource constrained project scheduling problem, 4OR 10 (1) (2012) 105–106.

[32] A. Lova, P. Tormos, F. Barber, Multi-mode resource constrained project scheduling: scheduling schemes, priority rules and mode selection rules, Intell. Artif. 30 (2006) 69–86.

[33] A. Lova, P. Tormos, M. Cervantes, F. Barber, An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes, Int. J. Prod. Econ. 117 (2) (2009) 302–316.

[34] M. Lundy, A. Mees, Convergence of an annealing algorithm, Math. Program. 34 (1986) 111–124.

[35] M. Mika, G. Waligóra, J. Weglarz, Simulated annealing and tabu search for multi-mode resource constrained project scheduling with positive discounted cash flows and different payment models, Eur. J. Oper. Res. 164 (2005) 639–668.

[36] E. Nabipoor Afruzi, E. Roghanian, A.A. Najafi, M. Mazinani, A multi-mode resource-constrained discrete time–cost tradeoff problem solving using an adjusted fuzzy dominance genetic algorithm, Sci. Iran. 20 (3) (2013) 931–944.

[37] K. Nonobe, T. Ibaraki, Formulation and Tabu Search Algorithm for the Resource Constrained Project Scheduling Problem (RCPSP), Technical Report, Kyoto University, 2001.

[38] D.C. Paraskevopoulos, C.D. Tarantilis, G. Ioannou, Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm, Expert Syst. Appl. 39 (4) (2012) 3983–3994.

[39] E. Pinson, C. Prins, F. Rullier, Using tabu search for solving the resource constrained project scheduling problem, in: Proceeding of the 4th International Workshop of Project Management and Scheduling, Leuven, 1994, pp. 102–106.

[40] A. Rahimi, H. Karimi, B. Afshar-Nadjafi, Using meta-heuristics for project scheduling under mode identity constraints, Appl. Soft. Comput. 13 (4) (2013) 2124–2135.

[41] M. Ranjbar, An optimal NPV project scheduling with fixed work content and payment on milestones, Int. J. Ind. Eng. Prod. Res. 22 (3) (2011) 181–186.

[42] M. Ranjbar, B. De Reyck, F. Kianfar, A hybrid scatter-search for the discrete time/resource trade-off problem in project scheduling, Eur. J. Oper. Res. 193 (1) (2008) 35–48.

[43] A. Sprecher, A. Drexl, Solving multi-mode resource constrained project scheduling problems by a simple, general and powerful sequencing algorithm, Eur. J. Oper. Res. 107 (1998) 431–450.

[44] A. Sprecher, S. Hartmann, A. Drexl, An exact algorithm for the project scheduling with multiple modes, OR Spektrum 19 (1997) 195–203.

[45] G. Taguchi, Introduction to Quality Engineering, Asian Productivity Organization, Tokyo, 1986.

[46] M. Vanhoucke, D. Debels, The impact of various activity assumptions on the lead time and resource utilization of resource constrained projects, Comput. Ind. Eng. 54 (2008) 140–154.

[47] V. Van Peteghem, M. Vanhoucke, A genetic algorithm for the preemptive and non-preemptive multi-mode resource

constrained project scheduling problem, Eur. J. Oper. Res. 201 (2) (2010) 409–418.

[48] H. Zhang, C. Tam, H. Li, Multi-mode project scheduling based on particle swarm optimization, Comput. Aided Civ. Infrastruct. Eng. 21 (2006) 93–103.

[49] H. Zhang, H. Li, C.M. Tam, Particle swarm optimization for resource constrained project scheduling, Int. J. Project Manage. 24 (1) (2006) 83–92.

[50] G. Zhu, J. Bard, G. Tu, A branch-and-cut procedure for the multimode resource constrained project scheduling problem, J. Comput. 18 (3) (2006) 377–390.