A Hybrid of Tabu Search and Simulated Annealing Algorithms for Preemptive Project Scheduling Problem

Behrouz Afshar-Nadjafi(✉), Mehdi Yazdani, and Mahyar Majlesi

Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran
afsharnb@alum.sharif.edu, m_yazdani@qiau.ac.ir, mahyar_industrial_engineer@yahoo.com

Abstract. In this paper, the resource constrained project scheduling problem with preemption is studied in which fixed setup time is needed to resume the preempted activities. The project entails activities with finish-to-start precedence relations, which need a set of renewable resources to be done. A mathematical model is presented for the problem and a hybrid of Tabu Search (TS) and Simulated Annealing (SA) with tuned parameters is developed to solve it. In order to evaluate the performance of the proposed TS/SA a set of 100 test problems is applied. Comprehensive statistical analysis shows that the proposed algorithm efficiently solves the problem. Furthermore, the benefits of preemption with setup times and its justifiability is demonstrated numerically.

Keywords: Project scheduling · Simulated annealing · Tabu search · Preemption · Set up time

1 Introduction

The resource constrained project scheduling problem (RCPSP) is a challenging optimization problem because of its application and NP-hardness [1]. The objective of RCPSP is minimization of project duration preserving the precedence and resources constraints. There are many solution methods to solve the RCPSP [2–5]. In classic scheduling problems it is supposed that each activity once started, will be continued nonstop. Preemptive project scheduling problem addresses the problem which relax this constraint and lets activities to be preempted and resumed later. To solve the preemptive case of project scheduling problems we can find some algorithms in the literature [6–8].

Considering setup times, it is a common assumption in machine scheduling [9–11], while this is not true in the context of project scheduling. Setup is defined as preparedness of all perquisites for the accomplishment of an activity. The required time for this preparedness is named setup time. When set up time is considerably small in comparison with processing time of activity, set up time can be merged into processing time. However, when activities require relatively long setup times, formulating and solving the problem as a traditional RCPSP, may culminate in poor solutions especially when preemption is permitted [12].

Motivation of this work is modelling the setup times in the preemptive case and solving the model. In doing so, a mixed integer formulation is proposed for the preemptive RCPSP with setup times. We call this problem PRCPSP-ST. Then, an efficient hybrid of TS/SA is developed to solve this NP-hard problem. Finally, the proposed algorithm is evaluated to solve the PRCPSP-ST and effect of setup time on project duration is analyzed. The rest of the paper is organized as follows: in Sect. 2 the PRCPSP-ST is described and formulated. In Sect. 3 the steps of the proposed algorithm are explained. Section 4 is devoted to the experimental results and validation of the proposed TS/SA. Finally, conclusion of the paper in presented in Sect. 5.

2 Problem Description

In preemptive resource constrained project scheduling problem with setup times (PRCPSP-ST), each activity i is performed in a single mode with deterministic duration of d_i. There is a max number K of renewable resource types where each activity i requires r_{ik}^{ρ} units of renewable resource type k ($k = 1, \ldots, K$) per time unit. Availability of the renewable resource type k, R_k^{ρ}, is constant throughout the project. The project is represented in activity on node, AON, style by $G = \{N, A\}$ in which, N, denoted activities (nodes) and, A, denotes finish to start precedence relations (arcs). The activities are numbered from the dummy start activity 0 to the dummy end activity $n + 1$. When an activity i is preempted, a setup time ST_i is needed to resume the preempted activity. In modelling PRCPSP-ST, we assume that:

- Preemption of the activities is in discrete time points.
- A setup time is needed to resume a preempted activity.
- Duration of an activity contains the initial setup time.
- Setup times are known and constant.
- Each activity is restarted immediately after its setup.
- Setups require same resources as process of activities.

The objective of the PRCPSP-ST is minimization of the project duration. A feasible schedule S is defined by a vector of activities start times satisfying all perquisite relations and resources constraints. Let $f_{i,j}$ represents the finish time of j^{th} unit of activity i. Also let $f_{i,0}$ denotes start time of activity i. By defining $x_{ij} = 1$ if j^{th} unit ($1 \le j \le d_i - 1$) of an activity i is preempted; $x_{ij} = 0$ otherwise, PRCPSP-ST can be conceptually modelled as follows:

$$Min\, C_{max} = f_{n+1,0} \tag{1}$$

$$f_{i,d_i} \le f_{j,0}; \qquad for\ (i,j) \in A \tag{2}$$

$$f_{i,j-1} + 1 \le f_{i,j} - x_{i,(j-1)}(1 + ST_i); \qquad for\ \ i = 0, \ldots, n+1\ \ and\ \ j = 0, \ldots, d_i \tag{3}$$

$$f_{i,j} - x_{i,(j-1)}(1 + ST_i) \le f_{i,j-1} + 1 + Mx_{i,(j-1)}; \qquad for\ \ i = 0, \ldots, n+1\ andj = 0, \ldots, d_i \tag{4}$$

$$f_{0,0} = 0 \tag{5}$$

$$\sum\nolimits_{i \in S_t} r_{ik}^{\rho} \leq R_k^{\rho}; \qquad for\ k = 1, \ldots, K\ and\ t = 1, \ldots, f_{n+1,0} \tag{6}$$

$$x_{i,j} \in \{0, 1\},\ f_{i,j} \in Integer; \qquad for\ \ i = 0, \ldots, n+1\ and\ j = 0, \ldots, d_i \tag{7}$$

The objective in Eq. (1) minimizes the project duration. Equation (2) preserves the finish to start precedence relations. Equations (3) and (4) impose a setup time after any preemption. Parameter M is a big positive number. Equations (3) and (4) maintains the logical relation between x_{ij} and f_{ij}. Equation (5) guarantees that start activity 0 be started at time 0. Equation (6) take care of the renewable resources availability. S_t denotes the set of activities which are in progress or their setups are in progress at time interval $[t - 1, t]$. Equation (7) specifies that f_{ij} are integers, while x_{ij} are binary.

3 Proposed Hybrid TS/SA

In this work a hybrid algorithm based on Tabu Search (TS) and Simulated Annealing (SA) is developed to solve PRCPSP-ST. The proposed algorithm uses both advantages of TS and SA. Tabu search applies an intelligent local (http://en.wikipedia.org/wiki/Local_search_(optimization)) search procedure to iteratively move from one potential solution to an improved one by using memory structures that describe the visited solutions or user-provided sets of rules [13]. Simulated annealing is a random search method that is initially proposed by Kirkpatrick et al. [14]. SA algorithm starts by generating an initial solution and by initializing the temperature parameter T. Then, at each iteration a solution S' is randomly generated in the neighborhood of the current solution S and if it is an improvement upon the current solution, it replaces the current solution, else it replaces the current solution with a probability generally computed following the Boltzmann distribution:

$$p = \exp\left(-\frac{f(s') - f(s)}{T}\right) \tag{8}$$

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. Tabu search and simulated annealing are successfully applied to a noticeable number of project scheduling problems. Solution representation, starting solution and neighborhood generation and tabus are the basic elements of TS/SA.

3.1 Solution Representation

Random-key (RK) and activity-list (AL) are two important representations for solutions in project scheduling. It is proved that AL representation outperform the others [15]. Herein the AL representation is applied to encode a schedule and a revised version of serial schedule generation scheme (SSGS) followed by a double justification is used to decode the codes to schedules. An activity i with duration of d_i is replaced by d_i activities with duration of 1 and the same resource requirements as the original activity. Then a feasible solution is represented by an $N' = \sum_{i=1}^{n} d_i$ elements vector (I). In this structure, each unit $j = 1, \ldots, d_i$ of an activity i is successor of the previous unit $(j - 1)$.

$$I = (J^1, J^2, \ldots, J^{N'}) \tag{9}$$

When a feasible solution represented by the above mentioned vector obtained, the start times of all activities is determined by a revised SSGS followed by a double justification. The SSGS sequentially adds activities to the partial schedule till a complete feasible schedule is achieved. In each step, the first un-scheduled activity in the AL is selected and the first possible start time is devoted to it preserving precedence and resource constraints. In the revised SSGS applied in this work, setup time after preemptions is embedded.

The double justification is an improvement procedure with two steps which is implemented on a schedule generated by the revised SSGS. In the first step, except for the first and the last dummy activities all activities are shifted to the right in the schedule which culminates in a right active schedule; a schedule where no activity can be finished later without delaying some other activities or increasing the makespan. In the second step, except for the initial activity; all activities are shifted to the left which results to a left active schedule; a schedule where no activity can be started earlier without violating the precedence or resource constraints.

3.2 Starting Solution

An initial solution is constructed by a Greedy Randomized Adaptive Search Procedure (GRASP) which is a two phase iterative procedure: construction and improvement [16]. The construction mechanism consists of two main components: a dynamic constructive heuristic and randomization. A solution is constructed by adding one new element from a set of elements at a time. The next element is selected randomly from a *candidate list* (*CL*). *CL* contains the activities that have all their predecessors already scheduled. The elements are prioritized based on a heuristic criterion that gives them a rank as a function of their insertion benefits. The second phase is a local search, which may be a basic or an advanced technique.

3.2.1 Construction Phase

At each stage, starting from the partial schedule assembled thus far, the *CL* is calculated. For each activity $j \in CL$, a priority *cost(j)* is calculated which is duration of the schedule resulted by adding the activity j to the partial schedule assembled thus far. Then activities with the lowest *cost(j)* are filtered to restricted candidate list (*RCL*). Length of *RCL* is controlled by a parameter $0 \leq \alpha \leq 1$. An activity is selected from *RCL* at random and inserted to partial schedule. This procedure continues until a complete schedule is reached.

3.2.2 Local Search Phase

After constructing the greedy randomized solution, the local search is employed on solution using the following Insertion procedure. First, an integer a is randomly selected from set $\{1, \ldots, N'\}$. Let J_e^a denotes the last predecessor and J_w^a denotes the first successor of activity J^a in activity list of the current solution I. Then, an integer h different from a is randomly selected from set $\{e+1, \ldots, w-1\}$. Finally, the activity in position a is moved to position h. This operator preserves the feasibility of the new solution. Local search procedure is continued until a predetermined number of iterations *max_neighbor* is reached. After the local search is done, the fitness of neighbor schedules is calculated and if the best neighbor schedule is better than current schedule, it replaces the schedule.

3.3 Main Structure of the Proposed TS/SA

Starting from the initial feasible solution S generated by GRASP, number of *max_-subiteration* neighbor solutions is considered. Main structure of the proposed algorithm is based on SA, while neighborhood structure is based on TS. In regular TS, one must evaluate the objective for every element of the neighborhood $N(S)$ of the current solution. An alternative is to instead consider only a random sample $N'(S)$ of $N(S)$, thus reducing the computational effort. This sample must be large enough to get a better solution with a fair probability at the next search stage. In our implementation, the size of $N'(S)$ is set equal to the square root of the number of activities N'. Finally, the best neighbor solution generated by TS will be subject of acceptance criterion of SA. The choice of an appropriate cooling schedule is crucial for the performance of the SA. In proposed TS/SA a geometric law $T_{k+1} = \beta T_k; 0 \leq \beta \leq 1$ is used which corresponds to an exponential decay of the temperature. The procedure is continued until a predetermined number of schedules, *max_schedules*, are produced. We obtained good results by indexing the number of produced schedules to the size of the problem, i.e. use of the small number of produced schedules for small problems and large number of produced schedules for larger problems. Therefore after some trials to obtain reasonable results, we fixed the number of produced schedules limited to $100N'$.

3.4 Neighborhood Structure (Moves)

The neighbor generation operators (moves) utilized in TS/SA is defined as follows:

i. Insertion: As described in Subsect. 3.2.2.
ii. Swap: Two random integers, c and d are drawn from set $\{1, \ldots, N'\}$ with $c < d$. Then the positions of activities J^c and J^d in the activity list are exchanged. Also, some activities between these positions are shifted to left or right such that feasibility of resulting solution is preserved.

3.5 Tabu List

The tabu list is managed as follows: Whenever a feasible move performed, its reverse move is added to the tabu list and the oldest existing move is removed from the front of the list according to the First-in-First-out (FIFO) rule. All moves on the tabu list are forbidden. However, if a tabu move can generate a solution better than the best found so far its tabu status may be cancelled in the light of the aspiration criterion so that the algorithm can move to this solution.

3.6 Calibrating

Value of the meta-heuristics parameters and operators are crucial factors in their performance. Herein the Taguchi experimental design is applied to calibrate the parameters of the proposed TS/SA. The Taguchi method determines the optimal level of controllable factors and minimizes the effect of noise [17]. In the proposed GRASP, the factors that should be tuned are *RCL* control parameter, α, number of GRASP iterations, GRASPit, and number of neighbors, Nmax. A randomly generated problem with 30 non-dummy original activities and 102 sub-activities with duration of 1 is utilized for parameters tuning. Using MINITAB software version 16, a L9 orthogonal array design is applied. To obtain more reliable data each experiment executed 4 times and the best result is considered. Also, same selection and reproduction scheme is used for all 36 runs.

However, we used the Taguchi design to calibrate the number of neighbors (subiteration), length of Tabu list, initial temperature and cooling rate considering three levels for each of these parameters. With tuned values for GRASP parameters and using a L27 orthogonal array design, the randomly generated problem is considered again. The number of produced schedules limited to 10000 as stopping criterion. We found optimal levels of α, GRASPit and Nmax as 0.3, 20, and 10, respectively, while tuned values for number of neighbors (subiteration), length of Tabu list, initial temperature and cooling rate are 10, 0.3, 20 and 0.7, respectively (Fig. 1).

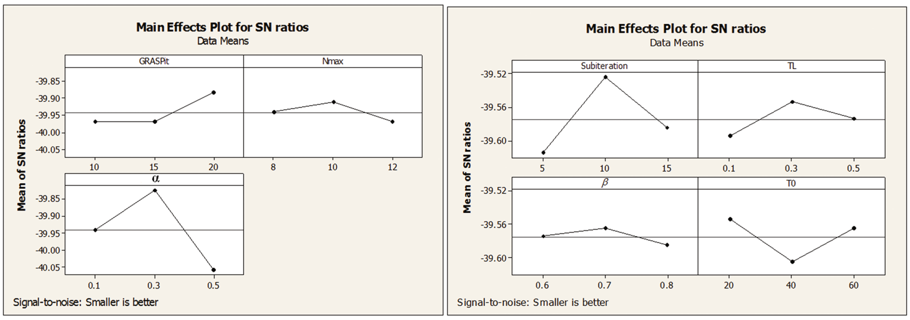

Fig. 1. The mean S/N ratio plot for the parameters

4 Performance Evaluation

4.1 Validation of Proposed Algorithm

In order to validate the proposed TS/SA algorithm for the PRCPSP-ST, a set of 10 problems with 10 non-dummy activities is generated by the generator ProGen developed by Drexl et al. [18] using the parameters given in Table 1.

Table 1. The parameter settings for the problem set

Control parameter	Value
Activity durations	Integer [1, 5]
Number of initial activities	Integer [1, 3]
Number of terminal activities	Integer [1, 2]
Maximal number of successors and predecessors	3
Number of renewable resources	2
Activity renewable resource demand (per period)	Integer [1, 10]
Resource factor (RF)	0.5
Resource strength (RS)	0.2
Network complexity (NC)	1.5

The proposed TS/SA were coded in Borland C++ 5.02 and executed on a personal computer with an Intel Core i5, 2.4 GHz processor and 4000 MB memory. Table 2 presents the computational results of the proposed algorithm. For problems with 10 activities, the results are compared with the optimal solutions obtained by LINGO 11. In Table 2, set up time ST of an activity is defined as a percent of its duration. Table 2 shows that when the number of activities is equal to 10, the results obtained by proposed TS/SA and LINGO are identical.

Also, Table 2 reveals that for problems 2, 3, 6, 8 and 10, makespan of the project in the preemptive RCPSP is same as the non-preemptive case. In problem 1, makespan of

Table 2. Comparison results for problems with 10 activities

Problem number	RCPSP	TS/SA for PRCPSP-ST			LINGO for PRCPSP-ST		
		ST = 0%	ST = 25%	ST = 50%	ST = 50%	ST = 50%	ST = 50%
Problem 1	14	13	14	14	14	14	14
Problem 2	19	19	19	19	19	19	19
Problem 3	15	15	15	15	15	15	15
Problem 4	18	17	17	18	18	18	18
Problem 5	20	19	19	20	20	20	20
Problem 6	19	19	19	19	19	19	19
Problem 7	27	26	26	26	26	26	26
Problem 8	22	22	22	22	22	22	22
Problem 9	24	23	23	23	23	23	23
Problem 10	19	19	19	19	19	19	19

PRCPSP without setup time (ST = 0%) is less than non-preemptive RCPSP, while when setup time is ST = 25% and 50%, preemption has no improving effect. In problems 4 and 5, a setup time up to ST = 25% and in problems 7 and 9, a setup time up to ST = 50% has improving effect on makespan of the project.

4.2 Experimental Results

In order to evaluate the proposed TS/SA for problems with more activities which LINGO is unable to solve the problem optimally in a reasonable time, a set of 90 project networks is considered. These project networks contain 30, 60 and 90 activities which are randomly chosen from the PSPLIB. The proposed TS/SA executed 10 times for each problem to obtain more reliable data. The results are reported in Table 3. In Table 3, Max.imp.(%) and Avr.imp.(%) denotes the maximum and average percentage of improvement in project makespan compared to non-preemptive RCPSP, respectively. Also, Imp.Inst.(%) denotes the percentage of improved problems compared to non-preemptive RCPSP. These measures in Table 3 reveal that when the number of activities or setup time is increased, the justifiability of preemption is reduced. This is

Table 3. Comparison of the results for problems with 30, 60 and 90 activities obtained by the TS/SA

ST(%)	Max.imp. (%)	Avr.imp. (%)	Imp.Inst. (%)	ARD (%)	Avr.CPU
0	6.4	1.18	30	0.08	163.586
25	4.7	0.34	10		106.549
50	2	0.18	6.66		77.648
0	8.1	0.82	16.66	0.19	1250.73
25	3.9	0.27	13.33		712.734
50	1.9	0.06	3.33		466.666
0	4.2	0.35	16.66	0.20	545.476
25	1.3	0.04	3.33		339.097
50	0.00	0.00	0.00		213.137

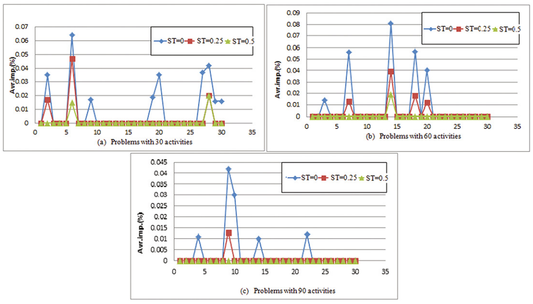

Fig. 2. Average improvement PRCPSP-ST to RCPSP

demonstrated in Fig. 2 based on Avr.imp.(%). Avr.CPU denotes the average CPU-time for the TS/SA (in seconds). Average CPU-time for TS/SA indicates when the number of activities is increased the complexity of problem is increased, too. Also, average CPU-time is a decreasing function of setup time ST%. ARD(%) denotes the average relative deviation percentages from the best found solution by the TS/SA. ARDs for the TS/SA algorithm are not high. This means that proposed TS/SA gives robust solutions. These results show that when the number of activities is large, while the LINGO is unable to solve the problem, there is a satisfying solution by the proposed TS/SA in a reasonable CPU time.

5 Conclusions

In this paper, we formulated and solved the preemptive resource constrained project scheduling problem with setup time to resume preempted activities. The objective is to schedule the activities in order to minimize of project duration subject to the precedence relations and renewable resource constraints. The problem was conceptually formulated, and then a hybrid of tabu search and simulated annealing (TS/SA) was designed to solve it. The parameters of proposed TS/SA were calibrated based on Taguchi experimental design. The evaluation of the proposed algorithm is done on 100 test problems with 10, 30, 60 and 90 activities. From the computation results, we found that the TS/SA algorithm could efficiently solve optimally the problems with 10 activities. Also, for problems with more activities which LINGO was unable to solve the problem optimally, we could find out that the proposed TS/SA is capable to find a satisfying solution in a reasonable CPU time. However results showed that the justifiability of preemption is a decreasing function of the number of activities and setup time.

References

1. Blazewicz, J., Lenstra, J., Rinnooy Kan, A.: Scheduling subject to resource constraints: classification and complexity. Discrete Appl. Math. **5**, 11–24 (1983)
2. Hartmann, S., Briskorn, D.A.: Surveys of variants and extensions of the resource constrained project scheduling problem. Eur. J. Oper. Res. **207**(1), 1–14 (2010)
3. Fang, C., Wang, L.: An effective shuffled frog-learning algorithm for resource constrained project scheduling problem. Comput. Oper. Res. **39**(5), 890–901 (2012)
4. Kone, O.: New approaches for solving the resource constrained project scheduling problem. 4OR **10**(1), 105–106 (2012)
5. Paraskevopoulos, D.C., Tarantilis, C.D., Ioannou, G.: Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm. Expert Syst. Appl. **39**(4), 3983–3994 (2012)
6. Van Peteghem, V., Vanhoucke, M.: A genetic algorithm for the preemptive and non-preemptive multi-mode resource constrained project scheduling problem. Eur. J. Oper. Res. **201**(2), 409–418 (2010)

7. Afshar-Nadjafi, B., Majlesi, M.: Resource constrained project scheduling problem with setup times after preemptive processes. Comput. Chem. Eng. **69**, 16–25 (2014)
8. Moukrim, A., Quilliot, A., Toussaint, H.: An effective branch-and-price algorithm for the preemptive resource constrained project scheduling problem based on minimal interval order enumeration. Eur. J. Oper. Res. **244**, 360–368 (2015)
9. Roshanaei, V., Naderi, B., Jolai, F., Khalili, M.: A variable neighborhood search for job shop scheduling with setup times to minimize makespan. Future Gener. Comput. Syst. **25**, 654–661 (2009)
10. Nagano, M.S., Silva, A.A., Lorena, L.A.N.: A new evolutionary clustering search for a no-wait flow shop problem with setup times. Eng. Appl. Artif. Intell. **25**(6), 1114–1120 (2012)
11. Liao, C.J., Chao, C.W., Chen, L.C.: An improved heuristic for parallel machine weighted flow time scheduling with family setup times. Comput. Math. Appl. **63**(1), 110–117 (2012)
12. Kolisch, R.: Project Scheduling Under Resource Constraints - Efficient Heuristics for Several Problem Classes. Physica, Heidelberg (1995)
13. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Norwell (1997)
14. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
15. Hartmann, S., Kolisch, R.: Experimental evaluation of state-of-the-art heuristics for the resource constrained project scheduling problem. Eur. J. Oper. Res. **127**, 394–407 (2000)
16. Pitsoulis, L.S., Resende, M.G.C.: Greedy randomized adaptive search procedure. In: Pardalos, P., Resende, M. (eds.) Handbook of Applied Optimization, pp. 168–183. Oxford University Press, Oxford (2002)
17. Taguchi, G.: Introduction to Quality Engineering. Asian Productivity Organization, Tokyo (1986)
18. Drexl, A., Nissen, R., Patterson, J.H., Salewski, F.: ProGen/πx - an instance generator for resource constrained project scheduling problems with partially renewable resources and further extensions. Eur. J. Oper. Res. **125**, 59–72 (2000)