Volume 2, Number 3, 2013

**International Journal of Research in Industrial Engineering**

journal homepage: www.nvlscience.com/index.php/ijrie

# Minimizing Total Resource Tardiness Penalty Costs in the Resource Constrained Project Scheduling Problem with Metaheuristic Algorithms

**R. Golestaneh**[1,*], **A. Jafari** [1], **M. Khalilzadeh**[2], **H. Karimi**[3]

[1]Department of Industrial Engineering, University of Science and Culture, Tehran, Iran.
[2]Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran.
[3]Department of Industrial Engineering, K.N.Toosi University of Technology, Tehran, Iran.

A R T I C L E  I N F O

*Article history* :
*Received*:
*March 24, 2013*
*Revised*:
*June 23, 2013*
*Accepted*:
*August 15, 2013*

***Keywords*** :
Resource constrained project scheduling,
Resource tardiness penalty costs,
Metaheuristic algorithms.

A B S T R A C T

In this paper, we study a resource-constrained project-scheduling problem in which the objective is minimizing total Resource Tardiness Penalty Costs. We assume renewable resources that are limited in number, are restricted to very expensive equipment and machines, therefore they are rented and used in other projects, and are not available in all project periods. In other words, there exists a predefined ready-date as well as a due date for each renewable resource type. In this way, no resource is utilized before its ready date. Nevertheless, resources are allowed to be used after their due date by paying penalty costs depending on the resource type. The objective is to minimize the costs of renewable resource usages. We formulated and mathematically modeled this problem as an integer-Linear programming model. Since our problem is NP-hard and also exact methods are only applicable in small scale, therefore metaheuristic methods are practical approaches for this problem; this means that metaheuristics are better for this problem. In order to authenticate the model and solution algorithm in small scale, we consider a network with low activity, and then solve the model of this network with both exact algorithms and SA-GA-TS metaheuristic algorithms. For more activities, as well as getting closer to the real world, we present a Simulated Annealing Algorithm to solve this problem. In order to examine the performance of this algorithm, data that had been derived from studied literature were used, and their answers were compared with Genetic Algorithm (GA) and Tabu Search Algorithm (TS). Results show that in average, quality of SA answers was better than those of the GA and TS algorithms. In addition, we use relaxation method to achieve an even higher validation for the SA algorithm. Finally all results in this paper indicate that both model and solution algorithm have high validity.

## 1. Introduction

Resource Constrained Project Scheduling Problem (RCPSP) is one of the most important issues in the areas of project scheduling and combinatorial optimization. RCPSP includes a project that has a number of specific activities with certain durations. RCPSP includes two constraints, precedence constraints in which technical precedence relation between activities is finish to start, and the other, resource constraints. The objective here is to minimize project

[*]Corresponding author
E-mail address: R.golestaneh@usc.ac.ir

makespan. In RCPSP, While an activity is being executed, preemption is not permitted (i.e. once started, an activity should be continued until it is completed). It is shown in Blazewicz, Lenstra, and Rinnooy-Kan [1] that the RCPSP, as a job-shop generalization, is NP-hard in the strong sense. Several solution procedures have been presented in the literature. They can be classified into three categories: exact methods  such as works of Demeulemeester and Herroelen [2], Mingozi et al. [3], and Patterson et al. [4], which mainly make use of various branch-and-bound procedures; heuristic methods based on the serial and the parallel schedule generation schemes  of Boctor [5], Kolisch, and Drexl [6]. Finally, metaheuristic methods based on Tabu search  from Baar et al. [7], Nonobe and Ibaraki [8], simulated annealing of Bouleimen and Lecocq [9], Cho and Kim [10], and genetic algorithms form Alcaraz and Maroto [11], Alcaraz et al. [12], Hartmann [13], Mendes et al. [14], and Valls et al. [15]. Surveys on several other solution procedures can be found in works of Demeulemeester and Herroelen [16]. The RCPSP under minimization of total resource tardiness penalty costs is an applicable problem, and a modified version of the RCPSP in which all assumptions and constraints are maintained, but the objective function is different. Moreover, several exact, heuristic and metaheuristic algorithms are proposed for scheduling problems with objective function related to tardiness. Nadjafi and Shadrokh [17] developed a B&B algorithm for the weighted earliness–tardiness project-scheduling problem using generalized precedence relations. Liaw, Lin, Cheng, and Chen [18] developed a B&B algorithm for scheduling unrelated parallel machines for minimizing total weighted tardiness. Essafi, Yazid, and Dauzère-Pérès [19] proposed a genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. Bilge, Kiraç, Kurtulan, and Pekgün [20] developed a Tabu search algorithm for solving the parallel machine total tardiness problem. In addition, Bilge, Kurtulan, and Kiraç [21] presented a Tabu search algorithm for the single machine total weighted tardiness problem. Bianco, Dell'Olmo, and Speranza [22] referred to resources that can be assigned to only one activity at a time in dedicated form. However, in this article, we assume that only a few renewable resources exist including expert human resources with high skill levels, particular types of cranes, and tunnel boring machines that have to be leased from third party providers. Considering that these limited renewable resources are employed in other projects, there is a dictated ready-date as well as a due-date for each item, such that no resource can be accessible before its ready-date; however, these resources are allowed to be used after their due dates by paying penalty cost, depending on the resource type. Ranjbar et al. [23] studied this problem with single mode for each activity, and availability of one unit for each type of renewable resource, in which they used the exact method of "branch and bound" in order to solve the problem. The problem we studied here is a generalization of the problem introduced by Ranjbar et al., with the difference that, the assumption that only one unit of each resource type is available, and no activity needs more than one resource for execution has been removed. In addition, we used metaheuristic algorithm to solve the problem. Since our problem with this profile is unique and is introduced for the first time, and considering that the optimal solution could not be found in the literature review, therefore we used different validation approaches. The rest of this article is organized as follows. Problem modeling and formulation are provided in Section 2.

The computational Study is presented in Section 3. Finally, conclusions are given in Section 4.

## 2. Problem Modeling and Formulations

In this article, we introduce a resource-constrained project-scheduling problem with finish-to-start precedence relations among project activities, considering renewable resource tardiness penalty costs. For each project, n activities and R renewable resources are given. $R_k$ is the availability of each renewable resource. The duration of an activity i is given as di. Activity j requires $r_{jk}$ units of renewable resource k. Our model is presented using an activity-on-node (AON) network. Thus, there are two dummy activities, first activity and the last activity (0 and n+1), which denote start and end of the project, respectively. The dummy start and end activities have zero duration and zero resource consumption. It should be noted that for each renewable resource K,$\rho_k$,δk, and $p_k$ show the ready date, due date, and tardiness penalty cost of this renewable resource, respectively. In order to embed the resource release dates in the network, one dummy node corresponding to each resource k, k=1… R, is added to the project network. This node displays an activity with duration $\rho_k$ with no resource requirements, which is a direct successor of the start dummy activity and direct predecessor of every activity $i\in N_k$ where $N_k$ is a set of activities that need a number of renewable resources of $K\in R$ type for execution. Each type of limited renewable resource is leased for a fixed period, starting from its ready time, and ending with its due-date, and is not available before its ready time; however it can be used after its due-date provided a tardiness penalty cost is paid. All activities are ready at the beginning of the project, and no preemption is permitted. We define the problem parameters as follows:

n: Number of project activities

R: Number of renewable resources

$R_k$: Renewable resource k availability

$\rho_k$: Ready time of renewable resource k

$\delta_k$: Due date of renewable resource k

$P_k$: Tardiness penalty cost of renewable resource k for each period

$Pr_j$: Set of predecessors of activity j

$d_j$: Duration of activity j

$r_{jk}$: Renewable resource k requirement for executing activity j

$T_k$: Is the renewable resource *k* tardiness, determined by $T_k=max\ \{CPk-\delta_k,\ 0\}$, where $CP_k$ is the release time of resource *k* in the project and equal to $CP_k=max\ \{f_i\}i\in N_k$.

(Earliest) finish time that is shown with *fi*, $(fi=s_i+d_i)$, where $s_i$ is an integer and shows the start time of activity *i*. The integer linear programming model for this problem can be formulated as follows:

$$\min z=\sum_{k=1}^{R} P_k T_k \qquad (1)$$

*s.t.*

$CP_k \geq S_i + d_i \; i \in N_k \quad k=1,2,\ldots,R$ (2)

$T_k \geq CP_k - \delta_k \quad k=1,2,\ldots,R$ (3)

$T_k \geq 0 \quad k=1,2,\ldots,R$ (4)

$S_j - S_i \geq d_i \quad j=1,2,\ldots,n \quad i \in Prj$ (5)

$\sum_{j \in A(t)} r_{jk} \leq R_k \quad k = 1,2,\ldots,R$ (6)

$S_i \geq \rho_k \; i \in N_k \quad k=1,2,\ldots,R$ (7)

$S_i, CP_k, T_k \in N^+ \quad for\ i=0,1,\ldots,n+1$ (8)

In the above model, objective function (1) represents the minimization of the total weighted resource tardiness penalty costs. It should be noted that as the cost of leasing for each renewable resource is fixed, it does not need incorporation in the objective function. Constraint (2) shows that the release time of each resource is not less than the finish time of each activity, which requires that resource. Constraint sets (3) and (4) ensure that $T_k$ is equal to $max\ \{CP_k - \delta_k, 0\}$. Constraint (5) is the precedence constraint implying that start time of activity j must be after all its predecessors are finished. Constraint (6) is the renewable resource constraint, where *A(t)* is the set containing in-progress activities at time *t*. Constraint (7) makes the starting times of all activities greater than or equal to the release dates of their corresponding resources. Constraint (8) ensures that variables $S_{i,}$, $CP_k$ and $T_k$ are non-negative integers. For validation of the model, and validation of the proposed SA solution, we considered a small network as a case example. Table 1 shows the resource information, which in relation to our example with n=7 real activities, m=2 resources and the corresponding graph is depicted in Figure 1. In this figure, the number shown above each node indicates activity duration and the number (s) below indicates the resources required for activity execution. The nodes labeled α and β correspond to ready times of resources 1 and 2, respectively. We also solved the model of this example using exact solution as well as SA algorithm, which both of their objective functions are shown in Table 2. As we can see in this table, in order to evaluate SA's algorithm performance we solved the model relevant to this network using TS and GA, which resulted in a similar objective function value.

Table 1. Resource information of the example project.

| | Resource Availability | Ready date | Due date | Penalty cost | $N_k$ |
|---|---|---|---|---|---|
| Res 1 | 4 | 2 | 11 | 14 | {1,2,4,6} |
| Res 2 | 5 | 3 | 10 | 11 | {3,5,7} |

Table 2. Comparison between performance of exact and metaheuristic algorithms for small network

| Exact Solution | SA | TS | GA |
|---|---|---|---|
| 14 | 14 | 14 | 14 |

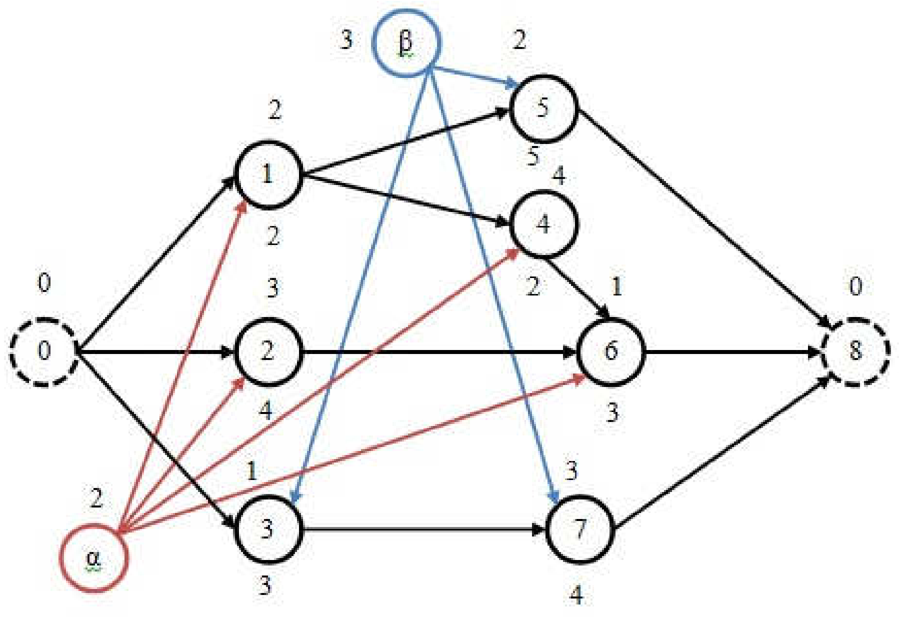

Figure 1. Network of the example project

## 3. Computational Study

The aim of this section is to evaluate the performance of SA approach by comparing it with GA and TS. The performance is appraised according to the objective function value quality. The solver applied in the study was MATLAB (2009) and cplex 12.2, run with a machine equipped with Windows 7, Intel (R) Core (TM) 2 with 2.53 GHz processor, and 3 GB of RAM.

### 3.1.Test problem

We used the sample problems library of PSPLIB [24] and selected two sets of project scheduling problems (i.e. j30 and 60). These data do not include some of our model parameters such as penalty cost of renewable resources, and ready time of renewable resources etc. hence; in this article we have employed discrete uniform distribution in selecting these parameters. The unit penalty costs of renewable resource tardiness were randomly chosen from discrete uniform distribution (10, 30). The ready times of renewable resources were randomly generated from discrete uniform distribution (0, 15), and the renewable resources due dates were randomly picked from discrete uniform distribution:

$$\sum_{j=1}^{n} t_j \Big/ 5, \; \sum_{j=1}^{n} t_j / 3$$

Moreover, the algorithms run in 20 iterations.

### 3.2.Performance of the Proposed Algorithm

In order to evaluate performance of the SA algorithm, the algorithm was implemented for the introduced test data. Obtained results were compared with results of the GA and TS algorithms. Comparison results are given in Tables (3) and (4), which are presented for j30 and j60 data, respectively. The first column of the table shows the Data Name. The ending section of the Data Name, which is in number form, represents the number of random parameter productions that do not constitute the data collection, and were randomly produced according to the previous section method. In fact, for each problem, parameters are randomly produced three-times. Columns 2, 3 and 4 show the name of the metaheuristic algorithm. Each of these columns comprises four other columns, in which the first column shows the minimum result obtained from five repetitions. Columns 2, 3, and 4 represent average, maximum, and standard deviation, respectively. The last row of this table displays result averages for simpler and better comparison. Figure (2) and Figure (3) represent performance difference between the three algorithms for the utilized data for j30 and j60 respectively. As can be seen, the vertical axis is objective function value, and the horizontal axis is data. In the next section, we will provide descriptions of obtained results.

Table 3. Comparison between performance of SA, TS and GA for J30

| Name | TS | | | | SA | | | | GA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Min | Avg. | Max | SDev | Min | Avg. | Max | SDev | Min | Avg. | Max | SDev |
| j301_1.1 | 563 | 563 | 563 | 0 | 563 | 563 | 563 | 0 | 563 | 563 | 563 | 0 |
| j301_1.2 | 515 | 515 | 515 | 0 | 515 | 515 | 515 | 0 | 515 | 515 | 515 | 0 |
| j301_1.3 | 289 | 289 | 289 | 0 | 289 | 289 | 289 | 0 | 289 | 289 | 289 | 0 |
| j305_2.1 | 1150 | 1178.53 | 1244 | 24.53 | 1150 | 1150 | 1150 | 0 | 1150 | 1157.08 | 1235 | 24.53 |
| j305_2.2 | 1234 | 1245.63 | 1275 | 8.64 | 1234 | 1234 | 1234 | 0 | 1234 | 1239.75 | 1257 | 10.40 |
| j305_2.3 | 1097 | 1112.42 | 1143 | 15.63 | 1034 | 1034 | 1034 | 0 | 1034 | 1041.75 | 1127 | 26.84 |
| j3010_1.1 | 733 | 733 | 733 | 0 | 733 | 733 | 733 | 0 | 733 | 733 | 733 | 0 |
| j3010_1.2 | 821 | 821 | 821 | 0 | 821 | 821 | 821 | 0 | 821 | 821 | 821 | 0 |
| j3010_1.3 | 538 | 538 | 538 | 0 | 538 | 538 | 538 | 0 | 538 | 538 | 538 | 0 |
| j3015_1.1 | 1109 | 1109 | 1109 | 0 | 1109 | 1109 | 1109 | 0 | 1109 | 1109 | 1109 | 0 |
| j3015_1.2 | 1117 | 1117 | 1117 | 0 | 1117 | 1117 | 1117 | 0 | 1117 | 1117 | 1117 | 0 |
| j3015_1.3 | 938 | 938 | 938 | 0 | 938 | 938 | 938 | 0 | 938 | 938 | 938 | 0 |
| Average | 842 | 846.63 | 857.08 | 4.07 | 836.75 | 836.75 | 836.75 | 0 | 836.75 | 838.47 | 853.5 | 5.15 |

Table 4. Comparison between performance of SA, TS and GA for J60

| Name | TS | | | | SA | | | | GA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Min | Avg. | Max | SDev | Min | Avg. | Max | SDev | Min | Avg. | Max | SDev |
| j601_1.1 | 547 | 564.43 | 587 | 12.95 | 547 | 547 | 547 | 0 | 547 | 554.5 | 572 | 12.07 |
| j601_1.2 | 204 | 212.52 | 276 | 23.63 | 195 | 195 | 195 | 0 | 195 | 198.7 | 210 | 6.32 |
| j601_1.3 | 486 | 498.52 | 524.69 | 18.35 | 486 | 486 | 486 | 0 | 486 | 495.2 | 532 | 19.39 |
| j605_1.1 | 1736 | 1826.46 | 2016 | 27.42 | 1616 | 1621.75 | 1639 | 10.40 | 1639 | 1765.30 | 1872 | 21.27 |
| j605_1.2 | 578 | 615.53 | 626 | 21.63 | 489 | 500.41 | 578 | 20.31 | 578 | 589.6 | 606 | 17.71 |
| j605_1.3 | 793 | 838.42 | 1153 | 42.74 | 649 | 658.83 | 767 | 34.06 | 837 | 934.53 | 1023 | 19.38 |
| j6010_1.1 | 1953 | 1989.30 | 2035 | 21.53 | 1903 | 1910.32 | 1926 | 21.42 | 1926 | 1954.3 | 1995 | 7.23 |
| j6010_1.2 | 758 | 758 | 758 | 0 | 758 | 758 | 758 | 0 | 758 | 758 | 758 | 0 |
| j6010_1.3 | 1133 | 1133 | 1133 | 0 | 1133 | 1133 | 1133 | 0 | 1133 | 1133 | 1133 | 0 |
| j6015_1.1 | 2177 | 2177 | 2177 | 0 | 2177 | 2177 | 2177 | 0 | 2177 | 2177 | 2177 | 0 |
| j6015_1.2 | 873 | 873 | 873 | 0 | 873 | 873 | 873 | 0 | 873 | 873 | 873 | 0 |
| j6015_1.3 | 1390 | 1390 | 1390 | 0 | 1390 | 1390 | 1390 | 0 | 1390 | 1390 | 1390 | 0 |
| Average | 1052.33 | 1073.01 | 1129.06 | 14.02 | 1018 | 1020.85 | 1039.08 | 7.18 | 1044.92 | 1068.59 | 1095.08 | 8.61 |

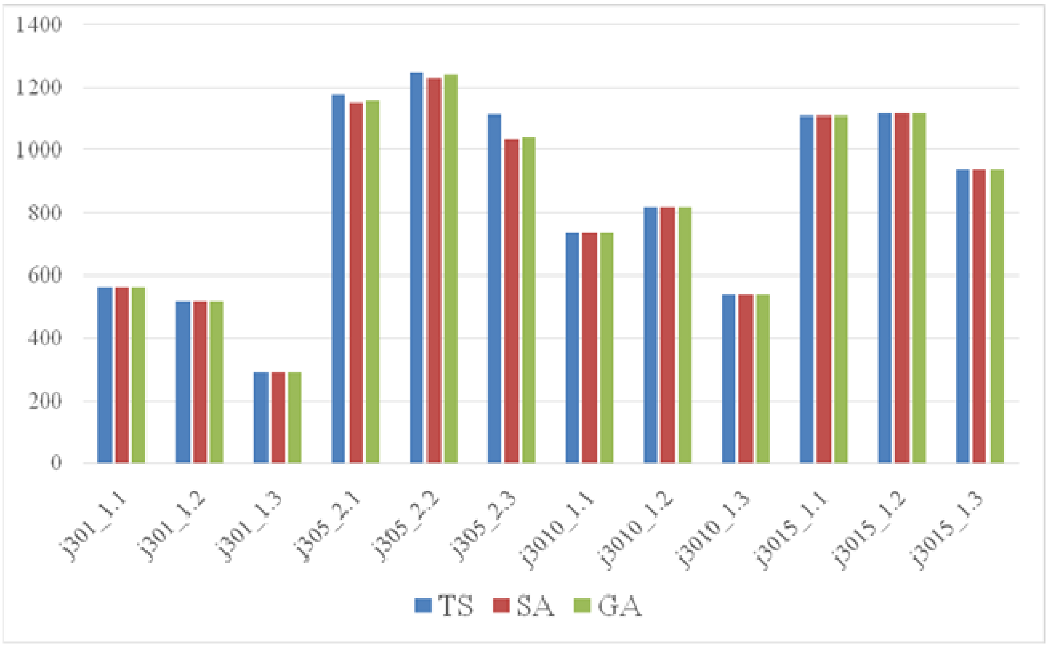

Figure 2. Comparison between performance of SA, TS and GA for J30

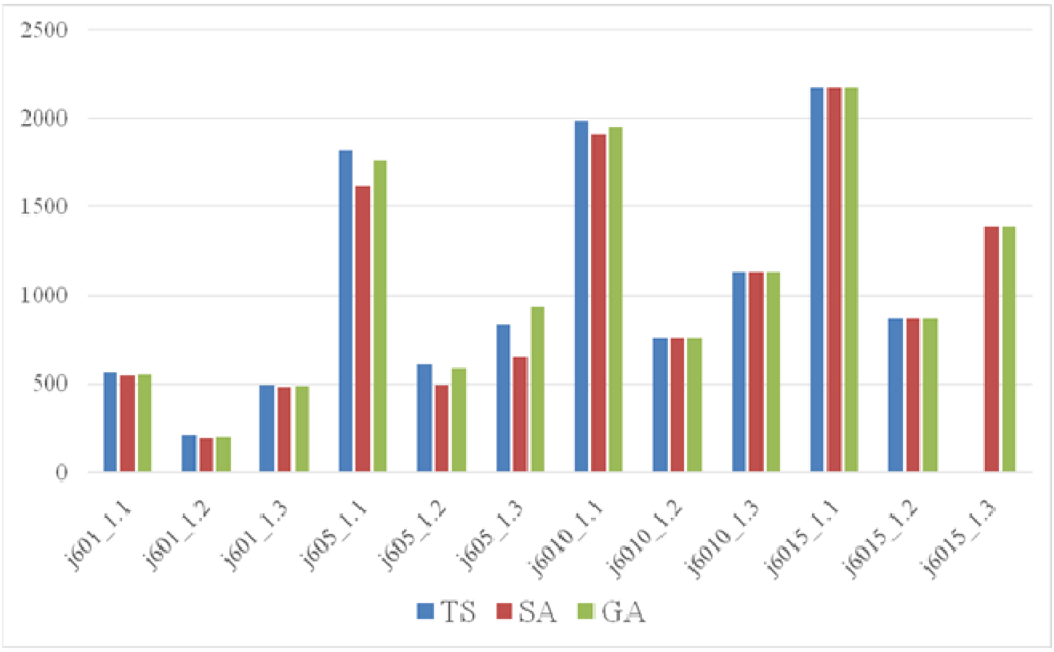

Figure 3. Comparison between performance of SA, TS and GA for J60

Table 5. Comparison between performance of exact and metaheuristic algorithms in relaxed state for J30

| Name | Relax Solution (exact) | TS | | | | SA | | | | GA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Min | Avg. | Max | SDev | Min | Avg. | Max | SDev | Min | Avg. | Max | SDev |
| j301_1.1 | 297 | 297 | 297 | 297 | 0 | 297 | 297 | 297 | 0 | 297 | 297 | 297 | 0 |
| j301_1.2 | 215 | 215 | 215 | 215 | 0 | 215 | 215 | 215 | 0 | 215 | 215 | 215 | 0 |
| j301_1.3 | 57 | 57 | 57 | 57 | 0 | 57 | 57 | 57 | 0 | 57 | 57 | 57 | 0 |
| j305_2.1 | 400 | 400 | 400 | 400 | 0 | 400 | 400 | 400 | 0 | 400 | 400 | 400 | 0 |
| j305_2.2 | 480 | 480 | 480 | 480 | 0 | 480 | 480 | 480 | 0 | 480 | 480 | 480 | 0 |
| j305_2.3 | 181 | 181 | 181 | 181 | 0 | 181 | 181 | 181 | 0 | 181 | 181 | 181 | 0 |
| j3010_1.1 | 666 | 666 | 666 | 666 | 0 | 666 | 666 | 666 | 0 | 666 | 666 | 666 | 0 |
| j3010_1.2 | 726 | 726 | 726 | 726 | 0 | 726 | 726 | 726 | 0 | 726 | 726 | 726 | 0 |
| j3010_1.3 | 445 | 445 | 445 | 445 | 0 | 445 | 445 | 445 | 0 | 445 | 445 | 445 | 0 |
| j3015_1.1 | 1109 | 1109 | 1109 | 1109 | 0 | 1109 | 1109 | 1109 | 0 | 1109 | 1109 | 1109 | 0 |
| j3015_1.2 | 1117 | 1117 | 1117 | 1117 | 0 | 1117 | 1117 | 1117 | 0 | 1117 | 1117 | 1117 | 0 |
| j3015_1.3 | 938 | 938 | 938 | 938 | 0 | 938 | 938 | 938 | 0 | 938 | 938 | 938 | 0 |
| Average | 552.58 | 552.58 | 552.58 | 552.58 | 0 | 552.58 | 552.58 | 552.58 | 0 | 552.58 | 552.58 | 552.58 | 0 |

Table 6. Comparison between performance of exact and metaheuristic algorithms in relaxed state for J60

| Name | Relax Solution (exact) | TS | | | | SA | | | | GA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Min | Avg. | Max | SDev | Min | Avg. | Max | SDev | Min | Avg. | Max | SDev |
| j601_1.1 | 433 | 433 | 433 | 433 | 0 | 433 | 433 | 433 | 0 | 433 | 433 | 433 | 0 |
| j601_1.2 | 171 | 171 | 171 | 171 | 0 | 171 | 171 | 171 | 0 | 171 | 171 | 171 | 0 |
| j601_1.3 | 486 | 486 | 486 | 486 | 0 | 486 | 486 | 486 | 0 | 486 | 486 | 486 | 0 |
| j605_1.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| j605_1.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| j605_1.3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| j6010_1.1 | 1903 | 1903 | 1903 | 1903 | 0 | 1903 | 1903 | 1903 | 0 | 1903 | 1903 | 1903 | 0 |
| j6010_1.2 | 736 | 736 | 736 | 736 | 0 | 736 | 736 | 736 | 0 | 736 | 736 | 736 | 0 |
| j6010_1.3 | 1133 | 1133 | 1133 | 1133 | 0 | 1133 | 1133 | 1133 | 0 | 1133 | 1133 | 1133 | 0 |
| j6015_1.1 | 2177 | 2177 | 2177 | 2177 | 0 | 2177 | 2177 | 2177 | 0 | 2177 | 2177 | 2177 | 0 |
| j6015_1.2 | 873 | 873 | 873 | 873 | 0 | 873 | 873 | 873 | 0 | 873 | 873 | 873 | 0 |
| j6015_1.3 | 1390 | 1390 | 1390 | 1390 | 0 | 1390 | 1390 | 1390 | 0 | 1390 | 1390 | 1390 | 0 |
| Average | 775.17 | 775.17 | 775.17 | 775.17 | 0 | 775.17 | 775.17 | 775.17 | 0 | 775.17 | 775.17 | 775.17 | 0 |

## 3.3.Discussion

Table 3 shows that the average mean of the results obtained using the Simulated Annealing algorithm for j30 equals 836.75, which is better than the obtained results by TS and GA algorithms, which are equal to 846.63 and 838.47 respectively. Its improvement values equal 9.88 and 1.72 respectively. This result also holds true for the average minimum and average maximum. Table 4 shows that the average mean of the results obtained by using the Simulated Annealing algorithm for j60, equals 1020.85, which is better than the obtained results by TS and GA algorithms, which are equal to 1073.01 and 1068.59, its improvement values equal 52.16 and 47.74 respectively .This result also holds true for the average minimum and average maximum. In addition, although by increasing the size of the problem and number of activities, the solution strength worsens than the state in which the number of activities was less. However, the Simulated Annealing algorithm in this case gives better answers than the TS and GA algorithms. Another important criterion for performance evaluation is standard deviation. As shown in table 3, the standard deviation average for the resulting answer from the TS and GA methods for j30 are4.07and 5.15 respectively. However, the standard deviation average for the resulting answer from our simulated annealing method equals zero. This result also holds true for j60. Therefore, the resulting answers of the Simulated Annealing is less scattered than the resulting answers of TS and GA algorithms, which is sign of the better accuracy of the later. As seen in Figure 2 and Figure 3, it is clear that the simulated annealing is more efficient than the TS and GA algorithms in both J30 and J60 .In these figures, as the gap between SA and other methods gets larger, indicates that the SA is more reliable than the TS and GA methods. Since our problem is NP-HARD, therefore, we relaxed the "resource constraint" to achieve higher validation for the solution method and proposed metaheuristic algorithm. Afterwards, we solved the problem using both exact solution and metaheuristic algorithms. As you can see in Tables 5 and 6, all the answers and results are identical. In fact, the results of the resource constraint relaxation are lower bounds of the problem.

## 4. Conclusion

In this paper, we studied the problem of minimizing total resource tardiness penalty costs in the resource constrained project-scheduling problem with metaheuristic algorithms. We formulated and mathematically modeled this problem as an integer- Linear programming model. Since our problem was NP-hard, we used metaheuristic algorithm as a solution procedure. At first, we considered a small network for validation of both model and proposed metaheuristic algorithm in small scale, and solved the model of this network with both exact solution and SA-GA-TS metaheuristic algorithms, the result of which showed that all objective function values were similar. Then we used a Simulated Annealing metaheuristic algorithm for the proposed project-scheduling problem. In order to confirm performance of the proposed algorithm in larger scale and closer to real world, the algorithm was applied to various test problems available in the literature, and reliability was compared with the Tabu Search (TS) algorithm and Genetic algorithm (GA). Computational results showed that the proposed algorithm provided competitive results in comparison with the TS and GA algorithms. Then we used relaxation method for higher validation of the solution method and proposed SA metaheuristic algorithm. This means that we relaxed the resource constraint, and afterwards obtained the objective function values using both exact method and metaheuristic algorithms for J30 and J60. Results showed that all the objective function values were identical. Finally, it should be noted that all the results of this paper indicates that both Model and Solution method are appropriately chosen.

## References

[1] Blazewicz, J., Lenstra, J. and Rinnooy-Kan, A. (1983), Scheduling subject to resource constraints – classification and complexity, *Discrete Applied Mathematics*, Vol. 5, pp. 11-24.

[2] Demeulemeester, E. and Herroelen, W. (1992), A branch-and-bound procedure for multiple resource-constrained project scheduling problem, *Management Science*, Vol. 38, pp. 1803-1818.

[3] Mingozi, A., Maniezzo, V., Ricciardelli, S. and Bianco, L. (1998), An exact algorithm for project scheduling with resource constraints based on a new mathematical formulation, *Management Science*, Vol. 44, pp. 714-729.

[4] Patterson, J.H., Slowinski, R., Talbot, F.B. and Weglarz, J. (1989), An algorithm for a general class of precedence and resource constrained scheduling problems, In: Sowinski, R., Weglarz, J. (Eds.), *Advances in project scheduling, Elsevier, Amsterdam*, pp. 3-28.

[5] Boctor, F.F. (1990), Some efficient multi-heuristic procedures for resource constrained project scheduling, *European Journal of Operational Research*, Vol. 49, pp. 3-13.

[6] Kolisch, R. and Drexl, A. (1996), Adaptative search for solving hard project scheduling problems, *Naval Research Logistics*, Vol. 43, pp. 23-40.

[7] Baar, T., Brucker, P. and Knust, S. (1998), Tabu-search algorithms and lower bounds for resource-constrained scheduling problem, In: Voss, S., Martello, S., Osman, I., Roucairol, C. (Eds.), Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, *Kluwer Academic*, pp. 1-18.

[8] Nonobe, K. and Ibaraki, T. (2002), Formulation and tabu search algorithm for the resource constrained project scheduling problem, In: Ribeiro, C.C., Hansen, P. (Eds.), Essays and Surveys in Metaheuristics, *Kluwer Academic Publishers*, pp. 557-588.

[9] Bouleimen, M. and Lecocq, H. (2003), A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple version, *European Journal of Operational Research*, Vol. 149, pp. 268-281.

[10] Cho, J.H. and Kim, Y.D. (1997), A simulated annealing algorithm for resource constrained project scheduling problems, *Journal of the Operational Research Society*, Vol. 48, pp. 735-744.

[11] Alcaraz, J. and Maroto, C. (2001), A robust genetic algorithm for resource allocation in project scheduling, *Annals of Operations Research*, Vol. 102, pp. 83-109.

[12] Alcaraz, J., Maroto, C. and Ruiz, R. (2004), *Improving the performance of genetic algorithms for RCPS problem*, In: Proceedings of the Ninth International Workshop on Project Management and Scheduling, Nancy, pp. 40-43.

[13] Hartmann, S. (1998), A competitive genetic algorithm for resource-constrained project scheduling, *Naval Research Logistics*, Vol. 456, pp. 733-750.

[14] Mendes, J.J.M., Goncalves, J.F. and Resende, M.G.C. (2009), A random key based genetic algorithm for the resource constrained project scheduling problem, *Computers and Operations Research*, Vol. 36, pp. 92-109.

[15] Valls, V., Ballestín, F. and Quintanilla, S. (2008), A hybrid genetic algorithm for the resource-constrained project scheduling problem, *European Journal of Operational Research*, Vol. 185, pp. 495-508.

[16] Demeulemeester, E. and Herroelen, W. (2002), *Project scheduling: a research handbook*, Kluwer Academic Publishers, Boston.

[17] Nadjafi, B.A. and Shadrokh, S. (2009), A branch and bound algorithm for the weighted earliness–tardiness project scheduling problem with generalized precedence relations, *Scientia Iranica,* Vol. 16, pp. 55-64.

[18] Liaw, C., Lin, Y., Cheng, C. and Chen, M. (2003), Scheduling unrelated parallel machines to minimize total weighted tardiness, *Computers & Operations Research,* Vol. 30, pp. 1777-1789.

[19] Essafi, I., Yazid, M. and Dauzère-Pérès, S. (2008), A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem, *Computers & Operations Research*, Vol. 35, pp. 2599-2616.

[20] Bilge, Ü., Kiraç, F., Kurtulan, M. and Pekgün, P. (2004), A tabu search algorithm for parallel machine total tardiness problem, *Computers & Operations Research*, Vol. 31, pp. 397-414.

[21] Bilge, Ü., Kurtulan, M. and Kiraç, F. (2007), A tabu search algorithm for the single machine total weighted tardiness problem, *European Journal of Operational Research*, Vol. 176, pp. 1423-1435.

[22] Bianco, L., DellOlmo, P. and Speranza, M.G. (1998), Heuristics for multimode scheduling problems with dedicated resources, *European Journal of Operational Research*, Vol. 107, pp. 260-271.

[23] Ranjbar, M., Khalilzadeh, M., Kianfar, F. and Etminani, K. (2012), 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.

[24] Kolisch, R. and Sprecher, A. (1997), PSPLIB - a project scheduling problem library, *European Journal of Operational Research*, Vol. 96, No. 1, pp. 205-216.