Assistant Professor Behrouz AFSHAR-NADJAFI
E-mail: afsharnb@alum.sharif.edu
Faculty of Industrial and Mechanical Engineering
Qazvin Branch, Islamic Azad University
Qazvin, Iran

USING GRASP FOR RESOURCE AVAILABILITY COST PROBLEM WITH TIME DEPENDENT RESOURCE COST

Abstract: *In this paper, we consider the resource investment project scheduling problem of minimizing the total resource availability costs by a given project deadline subject to recruitment and release dates for resources with time dependent recruitment setup cost. The project contains activities interrelated by finish-start type precedence relations with a time lag of zero, which require a set of renewable resources. A mixed integer programming formulation is proposed for the problem. The problem formed in this way is an NP-hard one forcing us to use Greedy Randomized Adaptive Search Procedure (GRASP) algorithm to obtain a satisfying solution. The performance of the proposed algorithm is evaluated on 150 test problems by statistically comparing in term of the objective function and computational times. Comparative computational results reveal that the proposed algorithm is efficient and effective.*

Keywords: *Project scheduling; Resource; investment; recruitment; release; GRASP*

JEL classification: C02, C52, C63

1. Introduction

Resource constrained project scheduling problem (RCPSP) is one of the main branches of project scheduling which is an NP-hard problem (Blazewicz et al., (1983)). 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 (1998), Hartmann and Kolisch (2000, 2006), Zhang et al. (2006), Jairo et al. (2010), Hartmann and Briskorn (2010), Agarwal et al. (2011), Fang and Wang (2012), Kone (2012) and Paraskevopoulos et al. (2012).

The resource availability cost problem (RACP) is a close variant of RCPSP which consists of scheduling the activities in a project such that the total cost of the renewable resources required completing the project by a pre-specified project deadline is minimized. This problem was introduced by Möhring (1984) as the resource investment problem. He proposes an exact procedure based on graph theoretical algorithms for comparability graph and interval graph recognition and orientation. Computational experiments were reported and the author showed that the problem is NP-hard. Rangaswamy (1998) proposed a branch-and-bound for the RACP and applied it to the same instance set used by Demeulemeester (1995). Drexl and Kimms (2001) proposed two lower-bound procedures for the RACP based on Lagrangian relaxation and column generation methods. Other exact procedures are proposed by Demeulemeester (1995) and Rodrigues and Yamashita (2010). There are some heuristic and meta-heuristic methods for the RACP in the literature: Yamashita et al. (2006) proposed a multi-start heuristic based on the scatter search methodology, Najafi and Niaki (2006) proposed a GA for a new RIP in which the goal was to maximize the discounted cash flows of the project payments. Shadrokh and Kianfar (2007) presented a genetic algorithm (GA) to solve the RACP when tardiness of the project is permitted with defined penalty costs, Ranjbar et al. (2008) developed a path relinking procedure and a genetic algorithm (GA), Najafi et al. (2009) proposed a parameter-tuned GA for the resource investment problem with discounted cash flows and GPRs, and Van Peteghem and Vanhoucke (2011) presented an artificial immune system algorithm for the problem.

In classic RACP, project activities have fixed duration, and need multiple constant amounts of renewable resources along their duration. Precedence constraints also exist. Given a deadline, the objective is to determine the start time of each activity and the capacity of each resource along the project execution time so that the total resource cost is minimized. It is assumed that resources (whether they are used or not) are assigned to the project for the total project duration and unit cost of each resource is fixed independent of its duration of availability. The assumption of the resources assignment to the project for the total project duration is one of the classical RACP shortcomings. It is likely that in reality, a certain resource is required only for a section of the project to accomplish some activities. However, the assumption of fixed unit cost of each resource is not true in practice. It is always unit cost of resources depends on their availability durations. Also, setup cost of resources depends on their recruitment time.

Therefore, the contribution of this paper is threefold: first, a mixed integer programming formulation is developed for the RACP problem which problem is minimizing the total cost of resources availabilities subject to recruitment and release dates for the resources with time dependent recruitment setup cost. We call this problem RACP-RR. In this problem setting, each resource is employed for a time horizon to accomplish some activities and after being accomplished; the resource is released. Also, cost of each resource depends of its availability duration and its recruitment time. Start time of each activity, recruitment and release date for each resource and capacity of each resource are decision variables. This model is not considered in the past literature. Second, a new efficient meta-heuristic solution procedure based on GRASP is developed for the problem due to NP-hardness of the problem. GRASP is a simple meta-heuristic that combines constructive and local search. In contrast to earlier research, we do not transform the problem to the RCPSP and a relatively new search algorithm will be used in this paper. Finally, we will analyze in this paper the effectiveness of the proposed method for the new version of RACP, statistically.

The paper is organized as follows: Section 2 describes the problem. Section 3 explains the basics of GRASP. In section 4 we explain the steps of our algorithm to solve the problem. Computational results are represented in section 5. Finally, section 6 contains the conclusions.

2. Problem description

The deterministic resource availability cost problem with recruitment and release dates for resources (RACP-RR) involves the scheduling of project activities without pre-emption on a set K of renewable resource types in order to minimize the total cost of the project's resources. Each activity i has a deterministic duration d_i $(1 \le i \le n)$ and requires r_{ik} units of resource type k $(1 \le k \le K)$. Let $\Omega(k)$ to be set of activities which require resource type k. The resource type k with constant availability R_k is employed from the first moment an activity from $\Omega(k)$ starts until the last activity in the $\Omega(k)$ finishes. We define c_k as the unit cost of resource k per time unit and A_{kt} as setup cost of resource type k if it is employed at time t.

In sequent, assume a project represented in *AON* format by a directed graph $G = \{N, A\}$ 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. Activities are not preempt-able and have one mode of execution. All parameters are deterministic and except c_k and A_{kt} $(k = 1,...,K \text{ and } t = 0,...,T)$ all of them are integral. A schedule S is defined by a vector of activity start times and is said to be feasible if all precedence and renewable resource constraints are satisfied.

The objective of the RACP-RR is to schedule a number of activities, in order to minimize the total cost of the resources availabilities subject to finish to start precedence relations with a time-lag of zero, and a fixed project deadline T. However, available resource capacities, recruitment and release dates for resources have to be determined. We have the following notations for RACP-RR:

n	number of activities
A	set of arcs of acyclic digraph representing the project
N	set of nodes of acyclic digraph representing the project
d_i	duration of activity i
EST_i	earliest start time of activity i (assuming infinite resource capacities)
LST_i	latest start time of activity i (assuming infinite resource capacities)
r_{ik}	resource requirement of activity i for renewable resource type k
K	Number of renewable resource(s)
T	deadline of the project
Z	objective function (total cost of the resources availabilities)

c_k	unit cost of resource type k per unit time
A_{kt}	setup cost of resource type k if it is employed at time t.
$\Omega(k)$	set of activities which require resource type k
s_i	start time of activity i (integer decision variable)
R_k	availability of resource type k (integer decision variable)
TS_k	recruitment date of resource type k (integer decision variable)
TF_k	release date of resource type k (integer decision variable)
X_{it}	1, if activity i starts at time t, 0, otherwise (binary decision variable)
Y_{kt}	1, if resource type k is employed at time t, 0, otherwise (binary decision variable)

By defining X_{it} as above, we have, $s_i = \sum_{t=EST_i}^{LST_i} tX_{it}$, (Pritsker et. al., (1969)). The variables X_{it} can only be defined over the time interval of the activity in question. These limits are determined using the traditional forward and backward pass calculations. The backward calculation is started from a fixed project deadline T.

Using the above notation, resource availability project scheduling problem with recruitment and release dates under the minimum total resource availabilities cost objective can be mathematically formulated as follows:

$$\min Z = \sum_{k=1}^{K} c_k R_k (TF_k - TS_k) + \sum_{t=0}^{T}\sum_{k=1}^{K} A_{kt} Y_{kt} \qquad (1)$$

subject to

$$\sum_{t=EST_j}^{LST_j} tX_{jt} \geq \sum_{t=EST_i}^{LST_i} tX_{it} + d_i \qquad \text{for all } (i,j) \in A \qquad (2)$$

$$\sum_{t=EST_i}^{LST_i} X_{it} = 1 \qquad \text{for } i \in N \qquad (3)$$

$$\sum_{i=2}^{n-1} \sum_{u=t-d_i+1}^{t} r_{ik} X_{iu} \leq R_k \qquad \text{for } k = 1,\ldots,K \text{ and } t = 0,\ldots,T \qquad (4)$$

$$TS_k \leq \sum_{t=EST_i}^{LST_i} tX_{it} \qquad \text{for } i \in \Omega(k) \text{ and } k = 1,\ldots,K \qquad (5)$$

$$TF_k \geq \sum_{t=EST_i}^{LST_i} tX_{it} + d_i \qquad \text{for } i \in \Omega(k) \text{ and } k = 1,\ldots,K \qquad (6)$$

$$\sum_{t=EST_n}^{LST_n} tX_{nt} \leq T \qquad (7)$$

$$TS_k = \sum_{t=1}^{T} tY_{kt} \qquad \text{for } k = 1,\ldots,K \qquad (8)$$

$$\sum_{t=0}^{T} Y_{kt} = 1 \qquad \text{for } k = 1,\ldots,K \qquad (9)$$

$$X_{it} \text{ and } Y_{kt} \in \{0,1\} \qquad \text{for } i \in N \text{ and } t = EST_i,\ldots,LST_i \qquad (10)$$

$$TF_k, TS_k, R_k \in \text{int}^+ \qquad \text{for } k = 1,\ldots,K \qquad (11)$$

The objective in Eq. (1) is to minimize the total cost of the project resources. The constraint set given in Eq. (2) imposes the finish-start precedence relations among the activities. Eq. (3) specifies that only one start time is allowed for each activity. Constraint set in Eq. (4) limits the total usage within each period to the available amount. Eq. (5) and (6) compute the recruitment and release dates for resources. Constraint in Eq. (7) guarantees that project deadline is not violated. Eq. (8) computes recruitment time of resource type *k*. Eq. (9) specifies that only one recruitment date is allowed for each resource type. Eq. (10) and (11) specifies that the decision variables X_{it} and Y_{kt} are binary, while TS_k and TF_k and R_k are integer. This formulation requires the definition of at most $2nT$ binary decision variables and of $3K$ integer variables. Also, the number of constraints of the formulation amounts to at most $|A| + n + KT + 2Kn + 2K + 1$.

3. GRASP

Greedy Randomized Adaptive Search Procedure (GRASP) is a simple metaheuristic that combines constructive and local search (Foe and Resende (1995), Pitsoulis and Resende (2002)). GRASP is an iterative procedure composed of two phases: solution construction and solution improvement. The best found solution is returned upon termination of the search procedure. The solution construction mechanism is characterized by two main ingredients: a dynamic constructive heuristic and randomization. Assuming that a solution *s* consists of a subset of a set of elements, the solution is constructed step-by-step by adding one new element at a time.

The choice of the next element is done by picking it uniformly at random from a *candidate list* (*CL*). The elements are ranked by means of a heuristic criterion that gives them a score as a function of the benefits if inserted in the current partial solution. The candidate list, called *restricted candidate list* (*RCL*), is composed of the best α elements. The heuristic values are updated at each step, thus the scores of elements change during the construction phase, depending on the possible choices. This constructive heuristic is called *dynamic*, in contrast to the *static* one which assigns a score to elements only before starting the construction. The basic structure of GRASP algorithm is presented in Table 1, where *s* is the current solution.

Table 1. GRASP Algorithm

```
While termination condition not met do
  s ← Construct Greedy Randomized Solution()
      Apply Local Search(s)
      Memorize Best Found Solution
End While
```

The length α of the restricted candidate list determines the strength of the heuristic bias. In the extreme case $\alpha = 1$ the best element would be added, thus the construction would be equivalent to a deterministic greedy heuristic. On the opposite, in case $\alpha = n$ the construction would be completely random. Therefore, α is a critical parameter which influences the sampling of the search space. GRASP construction mechanism is presented in Table 2, where the following notation is used:

s = the current solution
x = an element of solution
α = candidate list length
RCL= restricted candidate list

Table 2. Greedy randomized solution construction algorithm

```
s ← Φ
α ← Determine candidate list length()
While solution not complete do
    RCLα ← Generate restricted candidate list(s)
    x ← Select element at random(RCLα)
    s ← s ∪ {x}
        Update Greedy function(s)
End While
```

The second phase of the algorithm is a local search process, which may be a basic local search algorithm such as iterative improvement, or a more advanced technique. In GRASP the solution construction mechanism samples the most promising regions of the search space. This can be met by the choice of an effective constructive heuristic and an appropriate length of the candidate list. Also, the solutions constructed by the constructive heuristic belong to basins of attraction of different locally minimal solutions, because of choosing the constructive heuristic and the local search in a way such that they fit. The description of GRASP indicates that a basic GRASP does not use the history of the search process. The only memory requirement is for storing the problem instance and for keeping the best so-far solution. This is one of the reasons why GRASP is often outperformed by other metaheuristics. However, due to its simplicity, it is generally very fast and it can be successfully integrated into other search techniques.

4. Applying GRASP algorithm to RACP-PP

In this paper, a GRASP algorithm is designed to solve the mentioned above problem. In this regard, the steps of the proposed algorithm are briefly looked into, where the following notation is used:

N' = number of scheduled activities in partial schedule
T = deadline of project
N = number of project activities
q = index of iteration in *greedy randomized adaptive search procedure:*
m = index of iteration in *local search procedure*
α = GRASP control parameter

4.1. Greedy randomized adaptive search procedure

Initialization: set $q = 1$;

1. Set partial schedule $= \Phi$ and $N' = 0$;
2. Determine control parameter α ;
3. Create *RCL* (composed of the best α activities which are resource and precedence feasible to schedule);
4. Select an activity from *RCL* randomly, and insert it to partial schedule and put $N' = N' + 1$;
5. Update *RCL*, if *RCL* is not empty repeat step 4;
6. If *RCL* is empty and $N' < N$, (project deadline is violated) go to step 8;
7. If *RCL* is empty and $N' = N$, go to *local search procedure*;
8. Write "unfeasible schedule", set $q = q + 1$ and go to step 1;

4.2. Solution construction

Each solution is constructed from two sequences; resource capacities sequence and precedence feasible activity sequence. With considering infinite resource capacities, the starting precedence feasible activity sequence is generated using well-known, one-pass, priority-rule-based heuristics (Kolisch and Hartmann (1998)), which build a schedule by adding activities one by one. At each stage, starting from the partial schedule assembled thus far, a candidate list *CL*, sets of activities that are schedulable is calculated. Candidate list (*CL*) contains the activities that have all their predecessors already scheduled. For each activity $j \in CL$, a priority w_j is calculated and α activities with the highest priority are filtered to restricted candidate list (*RCL*). That means, *RCL* is composed of the best α activities which are precedence feasible to schedule. Then, we select an activity from *RCL*, at random. Selected activity is inserted to partial schedule with scheduling to start at earliest possible time according to precedence relations. This procedure continues until a complete schedule is reached. Six priority rules for the serial schedule generation scheme are proposed in (Kolisch and Hartmann (1998)): greatest rank positional weight (GRPW), latest finish time (LFT), latest start time

(LST), minimum slack (MSLK), most total successors (MTS) and shortest processing time (SPT). Of course, these rules are used is classic RCPSP with minimum makespan objective.

In RACP, four priority rules, i.e., weighting system (WS), random selection (RND), latest finish time (LFT) and random selection with probabilities proportional to LFT (RLFT) are proposed in shadrokh and kianfar (2007). Computational experiments were reported and the authors showed that the weighting system WS rule outperforms other rules. In our problem, we apply weighting system WS rule, proposed by shadrokh and kianfar (2007). This rule gives each activity of *CL* the chance of being selected proportional with its number of elements in the set of reachable nodes. On an activity on node network of a project, node *j* is reachable by node *i* (Schwindt, (1996)), if there is at least one directed path with origin *i* and destination *j*.

Applying serial schedule generation scheme we have an initial schedule. From this schedule and for each resource $k = 1,\ldots,K$ we can determine the amount of the resource, which is used in each unit of time, that is known as resource profile of current schedule and resource type *k*.

4.3. Local search procedure

After constructing the greedy randomized solution, the local search is employed on solution using the following procedure. In capacities sequence, a resource type *k*, is selected with probability:

$$P_k = \frac{c_k R_k (TF_k - TS_k) + \sum_{t=0}^{T} A_{kt} Y_{kt}}{Z} \tag{12}$$

and its availability R_k is replaced with R_k -1. For RACP-RR, the capacity of resource type *k* cannot be less than $\underline{R}_k = Max_{i=1,\ldots,n}\{r_{ik}\}$. Therefore, if $R_k < \underline{R}_k$, another resource type is selected. Let t_k be the earliest time instant for which amount of the resource used is more than R_k. All activities which are in progress on t_k in current schedule are identified. One activity *i* is selected among these activities at random. Subsequently, start time of activity *i*, s_i is replaced with $t_k + 1$ and all successors of activity *i* are shifted to the right if necessary so that the precedence constraints may not be violated. Finally, we can check possible right shift of activities which are completed before than t_k. This dominance rule can eliminate units of time which resource type *k* is completely useless if it is beneficial. Right shift dominance rule is demonstrated in Fig.1. It is clear that the resource is completely useless at unit time of 5 in Fig.1 (a). Right shift of activity 2 leads to resource profile in Fig.1 (b). Clearly, Right shift of activity 2 will be beneficial if $A_{k1} \le A_{k0}$ or $A_{k1} > A_{k0}$, $(A_{k1} - A_{k0}) < c_k R_k$. Generally, profitability of right shift depends on C_k, A_{kt} and R_k.

Identifying t_k (the earliest time instant for which amount of the resource used is more than R_k) this procedure continues until resource type *k* availability is met or project deadline is violated.

After the local search is done, the fitness of new schedule is calculated and if it is better than current schedule and project deadline is not violated, it replaces the schedule. Local search procedure is continued until a predetermined number of iterations (m_{max}) is reached.

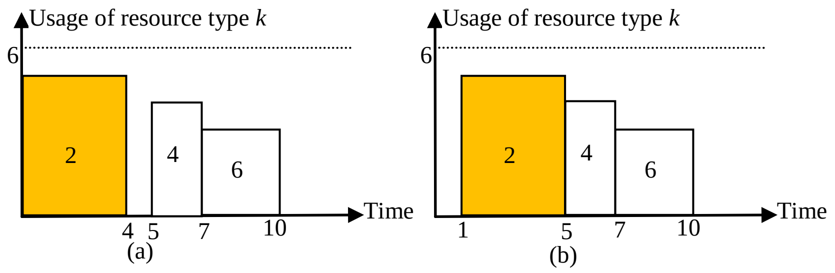

Figure 1. Right shift dominance rule for activity 2

4.4. Numerical example

In this section, we illustrate our method on a problem instance (Fig.2). There are 5 activities (and two dummy activities) and three resource types. The number above the node denotes the activity duration, while the numbers below the node denote the resource requirements, respectively. We assume the unit cost of resource types per unit time are c_1=2, c_2=3 and c_3=3. Also, we assume A_{1t}= A_{2t}= A_{3t}=10 for $t \leq 5$ and A_{1t}= A_{2t}= A_{3t}=12 for $t > 5$.
.

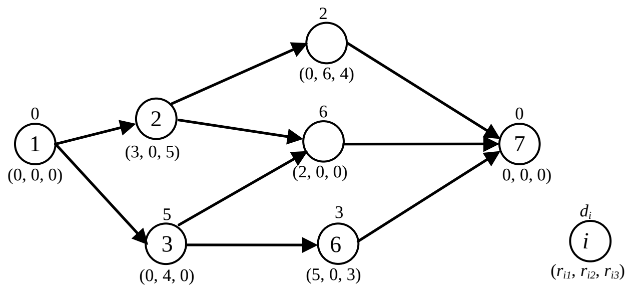

Figure 2. Problem instance for the RACP-RR

Using the Lingo version 11, based on branch and bound method, we obtained the optimal schedule with a cost of 441, R_1=5, R_2=6, R_3=5, TS_1=1 , TF_1=16 , TS_2=0 , TF_2=7 , TS_3=1 , TF_3=10 and S= (0, 1, 0, 5, 10, 7, 16). Resource profile of optimal schedule and resource types are presented in Fig.3. This optimal schedule is obtained by GRASP procedure, too.

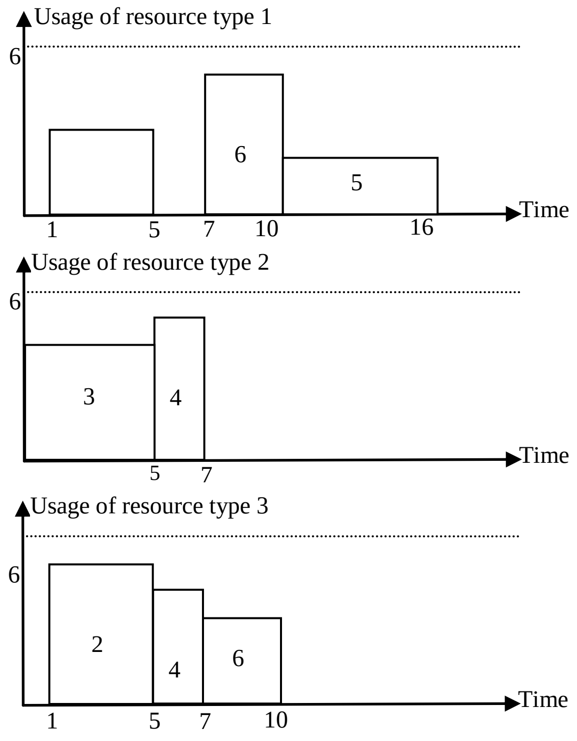

Figure 3. Optimal schedule to the problem example

5. Performance evaluation

5.1. The test problems

In order to validate the proposed GRASP algorithm for the RACP, a set of 150 problems was generated by the generator ProGen developed by Kolisch et al. (1995) using the parameters given in Table 3.

The indication [*x*,*y*] means that the real value is randomly generated on the interval [*x*,*y*]. Also, the indication {*x*,…, *y*} means that the integer value is randomly generated on the interval {*x*,…, *y*}. Resource availability is constant between recruitment and release dates. For each number of activities 30 problems were generated. The resource factor *RF* reflects the average portion of resource required per activity. The problem set was extended with unit cost of resource per unit time for each resource type which are randomly generated between 1 and 10. The deadlines were generated in the same way as described by Shadrokh and Kianfar (2007) by setting $T = 1.5 * EST_n$.

Table 3. The parameter settings for the problem set

Control Parameter	Value
Number of activities (non-dummy)	20, 30, 40,60,90
unit cost of resource per unit time for all resource types	[1,10]
setup cost of resource type *k* if it is employed at time *t*	[10, 100]
Activity durations	{1,…,10}
Number of initial activities	3
Number of terminal activities	3
Maximal number of predecessors	3
Maximal number of successors	3
Coefficient of network complexity (*CNC*)	1.5
Resource factor (*RF*)	0.7
Number of resource types	2, 3, 4
Activity resource (per period) demand for all resource types	{1,…,10}

5.2. Parameters setting

The performance of meta-heuristic algorithms depends excessively on the value of their parameters. In this paper, GRASP control parameter value is selected through the computational experiments. CPU-time limit was specified as a stopping criterion. 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. Therefore after some trials to obtain reasonable results, we fixed the CPU-time limit to 50 milliseconds per activity. The experiments indicated that the best value for GRASP control parameter α and number of iteration m_{max} , is as follows (Table 4):

Table 4. The tuned values for α and m_{max}

#activities	20	30	40	60	90
α	3	4	6	7	11
m_{max}	8	10	16	24	37

5.3. Experimental results

The proposed GRASP were coded in Borland C++ 5.02 and executed on a personal computer with an Intel Core2Dou, 2.5GHz processor and 3GB memory. Table 5 present 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 the GRASP if Lingo is not able to solve the problem). Proposed GRASP 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 sec.
NPM: Number of runs of problems for which GRASP was able to find optimum solution.
ACNT- Lingo: Average convergence time for Lingo (in seconds).
ACNT-GRASP: Average convergence time for the GRASP (in seconds).
ARD: Average relative deviation percentages.

Table 5. Computational results of the Lingo and GRASP

# Activities	# Resource types	#Problems	Lingo		GRASP		
			NPO	ACNT-Lingo	NPM	ARD	ACNT-GRASP
20	2	10	10	0.044	100	0.00%	0.096
20	3	10	10	0.112	100	0.00%	0.121
20	4	10	10	0.352	100	0.00%	0.130
30	2	10	10	2.967	100	0.00%	0.139
30	3	10	10	20.121	92	0.24%	0.158
30	4	10	10	61.253	89	0.32%	0.276
40	2	10	10	109.410	100	0.00%	0.293
40	3	10	7	198.728	90	0.46%	0.388
40	4	10	5	277.217	81	0.54%	0.456
60	2	10	0	___	93	0.52%	0.934
60	3	10	0	___	89	0.58%	1.521
60	4	10	0	___	77	0.61%	1.576
90	2	10	0	___	87	0.65%	3.053
90	3	10	0	___	78	0.74%	3.759
90	4	10	0	___	68	0.87%	3.907

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

$$RD = \frac{f - f^*}{f^*} \tag{13}$$

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

From Table 5 it can be observed that when the number of activities is less than or equal to 30, all 60 problems can be solved to optimality by Lingo within the allowed time limit. Also, Table 5 shows that when number of activities is greater than 30, while there are many instances that the Lingo is unable to solve, there is a solution by the proposed GRASP. Consequently, Lingo obtained optimum solutions for 82 out of 150 problems in 1000 seconds and GRASP algorithm solved all problems with low relative deviation and in a very shorter time (50 milliseconds per activity). Average CPU-time for Lingo indicates when the number of resource types is increased the complexity of problem is increased. ARD for the algorithm shows that proposed algorithm gives robust solutions. Also, NPM for the algorithm indicates that too many of executions of problems reach to the optimum solution.

6. Conclusions

In this paper, we attempted to solve the resource investment project scheduling problem with recruitment and release dates for resources (RACP-RR) with time dependent recruitment setup cost. In this problem the objective is to schedule the activities in order to minimize the total cost of the resources availabilities subject to the precedence constraints and a fixed deadline on project. This problem has not been studied ever before. The problem described with an integer programming model, and then the Greedy Randomized Adaptive Search Procedure (GRASP) proposed to solve it. The performance of the proposed algorithm on 150 test problems was compared with the results of the Lingo 11. From the computation results, we could clearly see that the GRASP algorithm could efficiently solve the project scheduling problem.

This research helps mangers to schedule their projects in order to minimize total resource costs in project environments when there is recruitment and release dates for resources with time dependent recruitment setup cost**.** Extensions of this research might be of interest, such as studying the RACP-RR combined with other project scheduling problems such as generalized precedence relation, multi-mode project scheduling, preemptive project scheduling and etc.

REFERENCES

[1] **Agarwal, A., Colak, S., & Erenguc, S. (2011),** ***A Neurogenetic Approach for the Resource-constrained Project Scheduling Problem;*** *Computers and Operations Research,* 38(1), pg.44-50;

[2] **Blazewicz, J., Lenstra, J., & Rinnooy Kan, A. (1983),** ***Scheduling Subject to Resource Constraints: Classification and Complexity;*** *Discrete Applied Mathematics,* 5, pg.11-24;

[3] **Demeulemeester, E. (1995),** ***Minimizing Resource Availability Costs in Time-limited Project Networks;*** *Manag. Scien.* 41, pg.1590-1598;

[4] **Drexl, A., & Kimms, A. (2001),** ***Optimization Guided Lower and Upper Bounds for the Resource Investment Problem;*** *Journal of Operations Research Society,* 52, pg.340-351;

[5] **Fang, C., & Wang, L. (2012),** ***An Effective Shuffled Frog-leaping Algorithm for Resource-constrained Project Scheduling Problem;*** *Computers and Operations Research,* 39(5), pg.890-901;

[6] **Feo, T. A., & Resende, M. G. C. (1995),** ***Greedy Randomized Adaptive Search Procedures;*** *Journal of Global Optimization,* 6, pg.109-133;

[7] **Hartmann, S., & Briskorn, D. A. (2010),** ***Survey of Variants and Extensions of the Resource-constrained Project Scheduling Problem;*** *European J.of Operational Research,* 207(1), pg.1-14;

[8] **Hartmann, S., & Kolisch, R. (2000),** ***Experimental Evaluation of State-of-the-art Heuristics for the Resource-constrained Project Scheduling Problem;*** *European Journal of Operational Research,* 127, pg.394-407;

[9] **Hartmann, S., & Kolisch, R. (2006),** ***Experimental Investigation of Heuristics for Resource-constrained Project Scheduling: An update;*** *European Journal of Operational Research,* 17, pg.23-37;

[10] **Jairo, R., Torres, M., & Edgar, G. F., Carolina Pirachicán-Mayorga (2010),** ***Project Scheduling with Limited Resources using a Genetic Algorithm;*** *International Journal of Project Management, 28*(6): pg.619-628;

[11] **Kolisch, R., & Hartmann, S. (1998),** ***Heuristic Algorithms for the Resource-constrained Project Scheduling Problem: Classification and Computational Analysis.*** *In: Weglarz, J. (Ed.), Project Scheduling: Recent Models, Algorithms and Applications; Kluwer Academic Publishers,* pg.147-178;

[12] **Kolisch, R., Sprecher, A., & Drexl, A. (1995),** ***Characterization and Generation of a General Class of Resource-constrained Project Scheduling Problems;*** *Management Science,* 41, pg.1693-1703;

[13] **Koné, O. (2012),** ***New Approaches for Solving the Resource-constrained Project Scheduling Problem;*** *4OR; 10*(1), pg.105-106;

[14] **Möhring, R. F. (1984),** ***Minimizing Costs of Resource Requirements in Project Networks Subject to a Fixed Completion Time;*** *Operations Research,* 32, pg.89-120;

[15] **Najafi, A. A., & Niaki, S. T. A. (2006), *A Genetic Algorithm for Resource Investment Problem with Discounted Cash Flows;*** *Applied Mathematics and Computations*, 183, pg.1057-70;

[16] **Najafi, A. A., Niaki, S. T. A., & Shahsavar, M. A. (2009), *Parameter-tuned Genetic Algorithm for the Resource Investment Problem with Discounted Cash Flows and Generalized Precedence Relations;*** *Computers and Operations Research,*36, pg.2994-3001;

[17] **Paraskevopoulos, D. C., Tarantilis, C. D., & Ioannou, G. (2012), *Solving Project Scheduling Problems with Resource Constraints via an Event List-based Evolutionary Algorithm;*** *Expert Systems with Applications,* 39(4), pg.3983-3994;

[18] **Pitsoulis, L. S., & Resende, M. G. C. (2002), *Greedy Randomized Adaptive Search Procedure;*** *In Handbook of Applied Optimization, P. Pardalos and M. Resende, Eds. Oxford University Press*, pg.168-183;

[19] **Pritsker, A. A. B., Watters, L. J., & Wolfe, P. M. (1969), *Multi Project Scheduling with Limited Resources;*** *Management Science*, 16, pg.93-108;

[20] **Rangaswamy, B. (1998), *Multiple Resource Planning and Allocation in Resource-Constrained Project Networks;*** *Ph.D. Thesis, Graduate School of Business, University of Colorado.*

[21] **Ranjbar, M., Kianfar, F., & Shadrokh, S. (2008), *Solving the Resource Availability Cost Problem in Project Scheduling by Path Relinking and Genetic Algorithm;*** *Applied Mathematics and Computations,* 196, pg.879-888;

[22] **Rodrigues, S., & Yamashita, D. (2010), *An Exact Algorithm for Minimizing Resource Availability Costs in Project Scheduling;*** *European Journal of Operational Research, 206*, pg.562-568;

[23] **Schwindt, C. (1996), *Generation of Resource Constrained Scheduling Problems with Minimal and Maximal Time Lags;*** *Report WIOR-489, Universitat Karlsruhe;*

[24] **Shadrokh, S., & Kianfar, F. (2007), *A Genetic Algorithm for Resource Investment Project Scheduling Problem, Tardiness Permitted with Penalty;*** *European Journal of Operational Research, 181*, pg.86-101;

[25] **Van Peteghem, V., & Vanhoucke, M. (2011), *An Artificial Immune System Algorithm for the Resource Availability Cost Problem;*** *Flexible Service Manufacturing Journal*, DOI 10.1007/s10696-011-9117-0;

[26] **Yamashita, D., Armentano, V., & Laguna, M. (2006), *Scatter search for Project Scheduling with Resource Availability cost;*** *European Journal of Operational Research*, 169, pg.623-637;

[27] **Zhang, H., Li, H., & Tam, C. M. (2006), *Particle Swarm Optimization for Resource-constrained Project Scheduling;*** *International Journal of Project Management*, 24(1), pg.83-92.