World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:6, No:2, 2012

A Branch and Bound Algorithm for Resource Constrained Project Scheduling Problem subject to Cumulative Resources

A.Shirzadeh Chaleshtari, Sh. Shadrokh

***Abstract*—**Renewable and non-renewable resource constraints have been vast studied in theoretical fields of project scheduling problems. However, although cumulative resources are widespread in practical cases, the literature on project scheduling problems subject to these resources is scant. So in order to study this type of resources more, in this paper we use the framework of a resource constrained project scheduling problem (RCPSP) with finish-start precedence relations between activities and subject to the cumulative resources in addition to the renewable resources. We develop a branch and bound algorithm for this problem customizing precedence tree algorithm of RCPSP. We perform extensive experimental analysis on the algorithm to check its effectiveness and performance for solving different instances of the problem in question.

***Keywords*—**Resource constrained project scheduling problem, cumulative resources, branch and bound algorithm, precedence tree.

I. Introduction

RENEWABLE and nonrenewable resources have been vast studied in project scheduling literature. On the other hand, cumulative resources [1] are another type of resources that in spite of their wide usage in practical cases, have been studied much less theoretically. Material resources of a project that are procured according to a procurement plan during the project horizon are good examples of this type of resources. Neumann and Schwindt[1] introduced these resources and their practical applications. Reference[2] modeled a batch production scheduling problem in process industry as a project scheduling problem subject to this type of resources while considering several extra considerations such as minimum and maximum time lags between activities. Reference[3] modeled a production scheduling problem using this resource type. Reference[4] modeled scheduling of tests in R&D projects in automotive industry as a customized model of *resource investment problem (RIP)*, aiming at minimization of the number of cumulative resources that are needed. Extra considerations such as activities ready times were made as well.

In this paper in order to study cumulative resource type more, we use the framework of a resource constrained project scheduling problem subject to cumulative resources (RCPSP-Cu) with finish-start precedence relations between activities and we develop a branch and bound algorithm for the problem.

A. Shirzadeh Chaleshtari is a Ph.D. candidate in the Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran (phone: 0098 912 605 7607; e-mail: a_shirzadeh@ie.sharif.edu).

Shahram Shadrokh is associate professor of the Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran (e-mail: shadrokh@sharif.edu).

In order that, we customize a branch and bound algorithm of resource constrained project scheduling problem (RCPSP).As we know, the literature on RCPSP is so extensive, dating back to 1960s, [5]. Many review papers such as [6], [7], [8], [9], [10], [11] and [12] summarize the literature on this problem. Due to the NP-hardness of this problem ([9],[13]), many studies have made on the problem developing inexact methods, such as heuristic methods of [14], [15], [16], [17], [18] and [19] and meta-heuristic methods of [20], [21],[22], [23], [24], [25], [26] and [27] based on genetic algorithm, [28], [29] and [30] based on simulated annealing, [31]and [32] based on tabu search and [33] based on ant colony. Besides, many exact algorithms such as binary programming based models of [34], [35] and [36], dynamic programming algorithm of [5] and branch and bound methods of [37], [38], [39], [40], [41], [42], [43] and [44] have been developed for the problem as well. Four basic approaches have been most used for the development of branch and bound methods for this problem, including precedence tree [45], extension alternatives [46], minimal delaying alternatives [39] and minimal forbidden sets [47]. Here we use precedence tree approach and develop our method for RCPSP-Cu based on this algorithm. So the rest of this paper is organized as the following. In the next section, RCPSP-Cu is described in detail. In section 3, the branch and bound algorithm for the problem is described. Section 4 is dedicated to computational analysis on and finally section 5 concludes the whole work.

II. Problem Description

A project with n non-dummy activities is considered. Finish-start precedence relations between activities are illustrated using an activity-on-node (AON) loop-less network, with dummy nodes 0 and n+1 as initial and terminal nodes respectively. $K = \{1, \dots, R\}$and$L = \{1, \dots, CR\}$ are the sets of renewable and cumulative resources respectively. Each activity i has a fixed duration of d_i and requires r_{ik} units of renewable resource k ($k \in K$) for each units of time over its duration and c_{ik} units of cumulative resource k ($k \in L$) which are used in the first period of its execution. Besides, each activity i (i=0,…,n+1) has a set of predecessor activities P_i. Each renewable resource k has a constant availability R_k over the project duration and each cumulative resource k has availability of CR_{kt} from the beginning of the project up to the period t ($CR_{kt} \geq CR_{k(t-1)}$). No preemption is permitted during the activities execution, all activities have just one mode and they are ready in the beginning of the project horizon. All parameters are deterministic and integrals. The problem is to

find the start time each of activity i, S_i, (i=0,…,n+1), such that all problem constraints are satisfied and the project makespan is minimized.

Suppose that the earliest and latest start time of each activity i, $EST_i$and LST_i, (i=0,…,n+1), is determined with forward and backward passes assigning $LST_{n+1}= LFT_{n+1}=T$, where T is an upper bound for optimum project makespan determined by any valid method. Also x_{it} be 1, if activity i starts at time t and 0 otherwise, then we have, $S_i = \sum_{t=EST_i}^{LST_i} t.x_{it}$ and the mathematical model of the problem is as the following:

$$Min \quad \left\{ \sum_{t=EST_{n+1}}^{lST_{n+1}} t.x_{(n+1)t} \right\} \tag{1}$$

$$\sum_{t=EST_j}^{LST_j} x_{jt} = 1, \qquad j = 1, \dots, n+1, \tag{2}$$

$$\sum_{t=EST_i}^{LST_i} (t+d_i)x_{it} \le \sum_{t=EST_j}^{LST_j} tx_{jt}, \qquad j = 1, \dots, n+1, \quad i \in P_j \tag{3}$$

$$\sum_{j=1}^{n} r_{jk} \sum_{\tau=t-d_j+1}^{t} x_{j\tau} \le R_k, \qquad k \in K = \{1, \dots, R\}, \quad t = 0, \dots, LST_J \tag{4}$$

$$\sum_{j=1}^{n} c_{jk} \sum_{\tau=EST_j}^{t} x_{j\tau} \le CR_{kt}, \qquad k \in L = \{1, \dots, CR\}, \quad t = 1, \dots, LST_J \tag{5}$$

$$x_{jt} \in \{0,1\}, \qquad j = 1, \dots, n+1, \quad t = EST_j, \dots, LST_j. \tag{6}$$

Objective (1) is to minimize the project makespan. Constraints (2) guarantee that each activity j can only has a single start time which has to be from the period $[EST_j,LST_j]$. Constraints (3) take into consideration precedence relations between each pair of activities (i,j) where i is an immediate predecessor of j. Constrains (4) and (5) regard renewable and cumulative resources usage limitations respectively and finally, constraints (6) denote the domain of variables.

III. Branch and Bound Algorithm

In this part we introduce a branch and bound algorithm for RCPSP-Cu problem. Fig. 1 shows the pseudo-code of the algorithm. In this algorithm, partial schedules containing parts of project activities are completed along the branching tree. In each level of the tree, a node not fathomed yet is selected for branching and according to the schedule in that node, one activity not scheduled yet is chosen. The selected activity is scheduled and a new node is generated. Relating to each node in which all activities have been scheduled, a feasible schedule is in hand for the problem. An upper bound for the project makespan is kept during the procedure which is equal to the makespan of the best feasible solution achieved up to the current time. Also a lower bound for the project makespan for each node in which only part of project activities have been scheduled is determined. For all new nodes generated in each step, this lower bound is compared to the current upper bound and if exceeds that, the node is fathomed. This procedure is repeated for all nodes not fathomed yet whenever the upper bound is modified in the algorithm. More details of the algorithm are described in the following.

- **Perform preprocessing and stop if the instance is infeasible**
- **Specify the initial upper bound**
- **Generate the initial node and select it for branching**
- **Branch selected node and fathom it**
- **Check new nodes for fathoming using dominance rules**
- **Fathom each new node not fathomed yet that contains feasible solution and update upper bound if necessary**
- **If upper bound has been updated after last branching:**
  a. **Perform fathoming check on all nodes not fathomed yet**

  Else:
  a. **Perform fathoming check on new nodes not fathomed yet**
- **If there is at least one node not fathomed yet:**
  a. **Select a new node for branching**
  b. **Continue from step 4**

  Else:
  a. **Report the best feasible solution achieved and stop**

Fig. 1 Pseudo-code of branch and bound algorithm for RCPSP-Cu

A. Preprocessing

There are two cases in which a given instance has no feasible solution, one is the case of renewable resources shortage in which at least one activity exists in given instance whose renewable resource requirement for at least one of the renewable resources is more than the availability of that resource. In this situation the activity cannot be executed and no feasible solution exists for the problem. The second case is the cumulative resource shortage in which at least one cumulative resource exists in the given problem whose total availability which is equal to the sum of all amounts ordered for that resource during the periods, is less than the total requirement of whole project activities for that resource. If none of these two states exists in a given instance, there are feasible solutions and the instance is feasible. So the branch and bound algorithm in the first step checks the feasibility of the instance.

B. Upper Bound Specification

Normally the tighter is the initial upper bound for an instance, the fasternodes are fathomed in the related branching tree. So in order to have a tight initial upper bound, first we

use a simple method of summing whole project activities durations plus the latest time all cumulative resources are ready for execution of entire project activities. Then we improve this upper bound using the following procedure. In this procedure we generate an activity list and schedule activities based on this list using a customized version of serial schedule generation scheme in which each selected activity is scheduled in the earliest feasible time that besides preserving precedence relations with other activities, enough renewable and also cumulative resources exist for execution of the activity. In order to generate a rather good activity list, we prioritize activities in decreasing order of the sum of latest start and finish times (LSTLFT) for activities. In [48] several most efficient heuristics for MRCPSP, embracing activities mode assignment and prioritization rules for scheduling using serial or parallel schedule generation schemes were compared and according to that, LSTLFT rule concluded the best among other single-pass heuristics for prioritizing activities.This method has been also compared to the best multi-pass methods. Results showed that although multi-pass methods need much more time than single-pass methods, they usually result in negligible improvement in the solution, so the LSTLFT choice seems the most appropriate among those compared considering time requirements.

In order to determine latest start time and finish time for activities, we use the simple initial upper bound we determined in our first step and we perform backward pass using this upper bound as the latest start time and finish time of the last project activity. So the initial upper bound determination for the problem is summarized as the pseudo-code shown in fig. 2.

- **Determine initial project upper bound IUB summing whole activities durations plus the latest time all cumulative resources are ready for execution of entire project activities**
- **Assign the last activity start time equal to IUB**
- **Determine the latest start time and finish time of activities using backward pass**
- **Compute LSTLFT of each activity and generate AL by prioritizing activities in non-decreasing order of their LSTLFTs**
- **Schedule activities based on the AL using customized serial schedule generation scheme for cumulative resources existence**
- **Report the finish time of last project activity as initial upper bound (IUB)**

Fig. 2 Pseudo-code of initial upper bound determination for RCPSP-Cu

During the algorithm, as the base procedure of branch and bound algorithm, we fathom any node in the last level of the branching tree which contains a feasible schedule for the problem. Then we substitute the current upper bound if the related objective function of the node is less than the current value of the upper bound.

C. Selecting Node for Branching

At the initial step in the first node, activity zero is scheduled at time zero. Then the initial node as the only available one is used for branching. In each other step of the algorithm, a node from the set of nodes not fathomed yet is selected for branching. In order to speed up the algorithm process, the back-tracking rule is used and from the set, a node is selected in which more activities have been scheduled. In this way we try to reach feasible solutions as fast as possible. In case a tie happens when more than one node exist in the set with maximum number of scheduled activities, the jump-tracking rule is used and one node is selected with the lower amount of lower bound for project makespan and if tie happens again, the node with the higher index is selected.

D. Branching

Once a node is selected for branching, all activities whose predecessors have been scheduled in the current node can be selected to be scheduled next. So regarding to each feasible activity that can be scheduled, one node is generated in the next level of the branching tree. The selected activity for each new node is scheduled in the earliest possible time in the way that first, its start time will not be earlier than finish time of any of its predecessors which have been all scheduled before and second, periodic availability of renewable and cumulative resources are enough for scheduling the activity in selected periods. Once the activity is scheduled, the periodic availability of all resources it uses is updated; availability of renewable resources from the start period of the activity until its finish period and availability of cumulative resources from the start period of the activity until the end of project are lessened as much as the related resource requirement of the activity. The process described here for scheduling any selected activity is similar to the scheduling each activity under serial schedule generation scheme which is described in part 4.4.1.

E. Dominance Rules

Here we use some dominance rules previously introduced in the literature for the precedence algorithm that implicitly help shorten the enumeration process. These rules which have been described in [49] for precedence tree algorithm of RCPSP can be extended to the customized algorithm for RCPSP-Cu. Experimental analysis in section 8 shows dramatic effectiveness of these rules in algorithm efficiency.

Regarding to each partial schedule in each node, we can specify an activity list that contains activities in the order they have been selected along the branching path. Sometimes it is possible to relate more than one activity list to a given partial schedule. For example for the partial schedule shown in fig. 3, we can have either activity lists (1,2,3,4) or (1,3,2,4). There can be one node associated to each of these activity lists in the branching tree. Therefore if we have two nodes in the branching tree with different activity lists whose related partial schedules are the same as each other, we can fathom one of them, as they contain similar solutions. Noting to this, if in any level g of the branching tree, the selected activity j_g is scheduled with the start time of less than the start time of one of the scheduled activities like i, we fathom the generated

node, because it contains the same partial schedule as another node whose activity list is the same as the current node but activities i and j_g are substituted with each other in the order. In other case, if recently selected activity j_g is scheduled with the same start time of the last activity in the schedule that has the latest start time, the node contains the same partial schedule with another node in which the order of two last activities are substituted. In this case, we can fathom one of the nodes and as a rule, in this case we fathom new node if j_g is less than the previously scheduled one with the same start time.

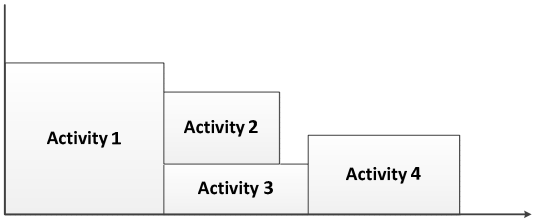

Fig. 3 Example of a partial schedule with two different activity lists

F. Lower Bound Determination for Nodes and Fathoming

According to the nodes present in the last level of the branching tree in which all activities have been scheduled, a feasible solution for the problem is in hand and therefore, the objective function can be assessed for each of these nodes based on their related makespan. But in the other nodes only a partial schedule of the project activities exists. So in order to determine a lower bound for the project makespan in such nodes, we use forward pass and specify project critical path length as a lower bound for the project makespan, however we modify the procedure of forward pass as the following by using extra information available for the solution. For activities present in partial schedule, we use the scheduled start time and finish time of the activity as their earliest start time (EST) and finish time (EFT) respectively. For activities not present in partial schedule, using the usual procedure, an earliest start time is specified equal to the latest finish time of all predecessors of the activity, we call this time *precedence based earliest start time (PBEST)*. The two dominance rules mentioned above can be used to tighten critical path based lower bound as well. According to them, the start time of each activity not scheduled yet like i cannot be less than the start time of the last scheduled activity like j if i>j and cannot be less than the start time of j plus one time unit if i<j, otherwise the related node is dominated and fathomed. So these rules are used in the determination of PBESTs.

Besides, a minimum for the activity start time is achievable based on the resource usage of activity as well, we call this minimum *resource based earliest start time (RBEST)*. So activity EST would be the maximum of the values PBEST and RBEST. In order to specify RBEST, we consider both renewable and cumulative resource usage of activity. As we described in part 3.4, we update the periodic availability of each resource along project planning horizon after scheduling each activity. So based on this updated resource availability data, RBEST is determined as the earliest period in which the availability of all resources either renewable or cumulative type are enough for execution of the activity. Of course regarding to the renewable resources, there should be enough availability for all duration of the activity, so all periods of duration must be checked for renewable resource availability but for cumulative resources, enough availability from the start period of the activity until the end of project planning horizon must be checked because the availability decreases as much as requirement for all these periods.

IV. Experimental Tests

In this part we present comprehensive experimental analysis on the branch and bound algorithm presented in this paper for RCPSP-Cu. Algorithm has been coded and executed on C#.NET 2010 platform on a PC with Core 2 Duo 2.53 GHz CPU and 3 GBs RAM.

A. Sample Problems

In order to have a full factorial design of parameters, sample problems of the project scheduling library (PSPLIB) [50] have been used. As RCPSP problem instances of the library are only subject to the renewable resources, MRCPSP problem instances have been used in order to have data of the nonrenewable resources as well. Using the following method, each MRCPSP instance has been transformed to a RCPSP-Cu instance:

- For each activity one mode is randomly chosen among the modes of that activity in the instance.
- Each nonrenewable resource k is substituted with a cumulative resource k in the instance and requirement of nonrenewable resource k for each activity is assumed as requirement of activity from cumulative resource k.
- For each cumulative resource k, the total amount required by all activities is randomly distributed between CPR first periods where CPR is the critical path length.

We use seven sets of multimode project scheduling problems from the PSPLIB, j10, j12, j14, j16, j18, j20 and j30 for experiments and convert them to RCPSP-Cu instances using the above procedure above.

B. Algorithm Validation

We use sample problems sets of j10, j16, j20 and j30 in this part and customize data of each instance so that its optimum objective function value is specified. In order that, for each instance we develop a random feasible solution schedule satisfying all constraints. Then if the makespan of the generated schedule is for example TM, we set one of the cumulative resource requirements of the last dummy activity equal to one and increase the input amount of that resource for period TM one unit. As the availability of the resource is equal to the activities requirement, the makespan of each solution cannot be less than TM up to which all the required amount of the resource is procured. So the feasible schedule in hand is an optimal solution.

We solve each revised instance as above with the branch and bound algorithm, limiting the solving time to 0.5 seconds. In order to assess the validity of algorithm, following quantity is computed regarding to each problem solved:

$$d = \frac{Z - Z_{Opt}}{Z_{Opt}} * 100 \quad (7)$$

Z: objective function value of the best solution achieved by branch and bound algorithm

Z_{Opt}: Optimal objective function value of the instance

The average and standard deviation of d for each problem set has been reported in table I. Results show little deviations of final solutions from the optimums

TABLE I
BRANCH AND BOUND ALGORITHM VALIDITY ASSESSMENT

Sample problems sets	Mean of d	Standard deviation of d
J10 (536 problems)	0.04	4.14
J16 (550 problems)	0.44	1.76
J20 (554 problems)	0.92	3.13
J30 (640 problems)	1.44	3.61

C. Algorithm Performance

In this part we use all sample problems set and from each of set, 50 instances are selected randomly. These instances are solved with the algorithm under three settings of limiting the solving times to 0.5, 3 and 10 seconds for each instance. In order to observe the effectiveness of two dominance rules used in the structure of the algorithm, we also solve all instances under three runtime settings using a simplified version of the branch and bound algorithm without these two dominance rules. Table 2, 3 and 4 show the number of problems solved from each set by original and simplified method under the related runtime limitation.

TABLE II
BRANCH AND BOUND METHOD PERFORMANCE ASSESSMENT-NUMBER OF PROBLEMS SOLVED TO OPTIMALITY-RUNTIME LIMITATION: 0.5 SECONDS

Sample problems sets	Simplified method	Original method
J10	49	50
J12	32	50
J14	4	46
J16	2	36
J18	2	27
J20	2	6
J30	0	0

TABLE III
BRANCH AND BOUND METHOD PERFORMANCE ASSESSMENT-NUMBER OF PROBLEMS SOLVED TO OPTIMALITY-RUNTIME LIMITATION: 3 SECONDS

Sample problems sets	Simplified method	Original method
J10	50	50
J12	50	50
J14	22	50
J16	3	47
J18	5	40
J20	2	25
J30	1	2

TABLE IV
BRANCH AND BOUND METHOD PERFORMANCE ASSESSMENT-NUMBER OF PROBLEMS SOLVED TO OPTIMALITY-RUNTIME LIMITATION: 10 SECONDS

Sample problems sets	Simplified method	Original method
J10	50	50
J12	50	50
J14	38	50
J16	5	49
J18	6	47
J20	2	39
J30	1	3

V. CONCLUSION

In this paper we studied resource-constrained project scheduling problem subject to cumulative resources (RCPSP-Cu). We developed a branch and bound algorithm for RCPSP-Cu based on the precedence tree approach and performed extensive experimental analysis on this algorithm. Validation assessment of the algorithm using revised instances with optimum solutions in hand showed algorithm validity via little deviation of the algorithm solutions from optimums. Checking the effectiveness of dominance rules used in the structure of the algorithm via comparing the performance of the original algorithm with its simplified version without dominance rules showed noticeable effectiveness of the rules. Finally performance assessment of the algorithm checking ability to solve instances of different sizes under different time limitations to optimality showed its efficiency for instances of size j18 and smaller for short runtime limitation of 0.5 seconds and j20 and smaller for more available runtimes of 3 and 10 seconds.

For further study of the problem in question, other exact algorithms can be developed and checked specially using ideas of other RCPSP branch and bound approaches. Also considering the NP-hardness of the problem, development of inexact and bounding algorithms would be useful.

REFERENCES

[1] K. Neumann, C. Schwindt, "Project scheduling with inventory constraints," *Mathematical Methods of Operations Research*, vol. 56, pp. 513–533, 2002.

[2] C. Schwindt, and N. Trautmann, "Batch scheduling in process industries: An application of resource-constrained project scheduling,"*OR Spectrum*, vol. 22, pp. 501–524, 2000.

[3] K. Neumann, C.Schwindt, and N. Trautmann, "Scheduling of continuous and discontinuous material flows with intermediate storage restrictions,"*European Journal of Operational Research*, vol. 165, pp. 495–509, 2005.

[4] J. H. Bartels, and J. Zimmermann, "Scheduling tests in automotive R&D projects,"*European Journal of Operational Research*, vol. 193, pp. 805–819, 2009.

[5] J.A. Carruthers, and A. Battersby, "Advances in critical path methods,"*Operational Research Quarterly*, vol. 17, pp. 359–380, 1996.

[6] O. Icmeli, S.S. Erenguc, and C.J. Zappe, "Project scheduling problems: a survey," *International Journal of Operations & Production Management*, vol. 13(11), pp. 80–91, 1993.

[7] L. Özdamar, and G.Ulusoy, "A survey on the resource constrained project scheduling problem,"*IIE Transactions*, vol. 27, pp. 574–586, 1995.

[8] W. Herroelen, E. Demeulemeester, and B. De Reyck, "Resource-constrained project scheduling: a survey of recent developments,"*Computers & Operations Research*, vol. 25, pp. 279–302, 1998.

[9] P. Brucker, A.Drexl, R.Mohring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: notation, classification, models, and methods,"*European Journal of Operational Research*, vol. 112, pp. 3–41, 1999.

[10] S. Hartmann, and R.Kolisch, "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, vol. 127, pp. 394–407, 2000.

[11] R. Kolisch, and R. Padman, "An integrated survey of deterministic project scheduling,"*OMEGA*, vol. 29, pp. 249–272, 2001.

[12] R. Kolisch, and S. Hartmann, "Experimental investigation of heuristics for resource-constrained project scheduling: an update,"*European Journal of Operational Research*, vol. 174, pp. 23–37, 2006.

[13] J. Blazewicz, J. Lenstra, and A. RinnooyKan, "Scheduling subject to resource constraints: Classification and complexity," *Discrete Applied Mathematics*, vol. 5, pp. 11–24, 1983.

[14] E.W. Davis, and J.H. Patterson, "A comparison of heuristic and optimum solutions in resource-constrained project scheduling,"*Management Science*, vol. 21, pp. 944–955, 1975.

[15] D.F. Cooper, "Heuristics for scheduling resource-constrained projects: an experimental investigation,"*Management Science*, vol. 22, pp. 1186–1194, 1976.

[16] G. Ulusoy, and L. Ozdamar, "Heuristic performance and network/resource characteristics in resource-constrained project scheduling,"*Journal of the Operational Research Society*, vol. 40, pp. 1145–1152, 1989.

[17] FF. Boctor,"Some efficient multi-heuristic procedures for resource-constrained project scheduling,"*European Journal of Operational Research*, vol. 49, pp. 3–13, 1990.

[18] L. Özdamar, and G.Ulusoy, "A local constraint based analysis approach to project scheduling under general resource constraints,"*European Journal of Operational Research*, vol. 79, pp. 287–298, 1994.

[19] R. Kolisch, "Efficient priority rule for the resource-constrained project scheduling problem,"*Journal of Operations Management*, vol. 14, pp. 179–192, 1996.

[20] V.J. Leon, and B. Ramamoorthy, "Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling,"*OR Spektrum*, vol. 17, pp. 173–182, 1995.

[21] J.K. Lee, and Y.D. Kim, "Search heuristics for resource-constrained project scheduling,"*Journal of the Operational Research Society*, vol. 47, pp. 678–689, 1996.

[22] S. Hartmann, "A competitive genetic algorithm for the resource-constrained project scheduling,"*Naval Research Logistics*, vol. 45, pp. 733–750, 1997.

[23] S. Hartmann "A self-adapting genetic algorithm for project scheduling under resource constraints,"*Naval Research Logistics*, vol. 49, pp. 433–448, 2002.

[24] J. Alcaraz, and C. Maroto,"A robust genetic algorithm for resource allocation in project scheduling,"*Annals of Operations Research*, vol. 102, pp. 83–109, 2001.

[25] K.S. Hindi, H. Yang, and K. Fleszar, "An evolutionary algorithm for resource-constrained project scheduling,"*IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 512–518, 2002.

[26] Y.C. Toklu, (2002). "Application of genetic algorithms to construction scheduling with or without resource constraints," *Canadian Journal of Civil Engineering*, vol. 29, pp. 421–429, 2002.

[27] V. Valls, F.Ballestin, and S. Quintanilla, "A hybrid genetic algorithm for the resource constrained project scheduling problem,"*European Journal of Operational Research*, vol. 185, pp. 495–508, 2008.

[28] FF. Boctor, "An adaptation of the simulated annealing algorithm for solving resource-constrained project scheduling problems,"*International Journal of Production Research*, vol. 34, pp. 2335–2351, 1996.

[29] J.H. Cho, and Y.D. Kim, "A simulated annealing algorithm for resource-constrained project scheduling problems,"*Journal of the Operational Research Society*, vol. 48, pp. 736–744, 1997.

[30] K. Bouleimen, and H.Lecocq, "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version,"*European Journal of Operational Research*, vol. 149, pp. 268–281, 2003.

[31] E. Pinson, C. Prins, and F. Rullier, "Using tabu search for solving the resource- constrained project scheduling problem," in *Proc. 4th international workshop on project management and scheduling*, Leuven, Belgium, 1994, pp. 102–106.

[32] P.R. Thormas, and S. Salhi, "A tabu search approach for the resource constrained project scheduling problem," *Journal of Heuristics*, vol. 4, pp. 123–139, 1998.

[33] D. Merkle, M.Middendorf, and H. Schmeck, "Ant colony optimization for resource-constrained project scheduling,"*IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 333–346, 2002.

[34] A.A.B. Pritsker, L.J. Watters, P.M. Wolfe, "Multiproject scheduling with limited resources: a zero-one programming approach," *Management Science*, vol. 16, pp. 93–107, 1969.

[35] J.H. Patterson, and W.D. Huber, "A horizon-varying, zero-one approach to project scheduling,"*Management Science*, vol. 20, pp. 990–998, 1974.

[36] J.H. Patterson, and G.W. Roth, "Scheduling a project under multiple resource constraints: a zero-one programming approach,"*AIIE Transactions*, vol. 8, pp. 449–455, 1976.

[37] E.W. Davis, and G.E. Heidorn, "An algorithm for optimal project scheduling under multiple resource constraints," *Management Science*, vol. 17, pp. 803–816, 1971.

[38] F.B. Talbot, and J.H. Patterson, "An efficient integer programming algorithm with network cuts for solving resource constrained scheduling problems," *Management Science*, vol. 24, pp. 1163–1174, 1978.

[39] N. Christofides, R. Alvarez-Valdes, and J.M. Tamarit, "Project scheduling with resource constraints: a branch and bound approach," *European Journal of Operational Research*, vol. 29, pp. 262–73, 1987.

[40] E. Demeulemeester, and W. Herroelen, "A branch-and-bound procedure for the multiple resource-constrained project scheduling problems,"*Management Science*, vol. 38, pp. 1803–1818, 1992.

[41] E. Demeulemeester, and W. Herroelen, "New benchmark results for the resource-constrained project scheduling problem," *Management Science*, vol. 43, pp. 1485–1492, 1997.

[42] P. Brucker, S.Knust, A.Schoo, and O. Thiele, "A branch and bound algorithm for the resource-constrained project scheduling problem,"*European Journal of Operational Research*, vol. 107, pp. 272–288, 1998.

[43] A. Mingozzi, V.Maniezzo, S. Ricciardelli, and L. Bianco, "An exact algorithm for project scheduling with resource constraints based on new mathematical formulation,"*Management Science*, vol. 44, pp. 714–729, 1998.

[44] U. Dorndorf, E. Pesch, and T. Phan-Huy, "A branch-and-bound algorithm for the resource-constrained project scheduling problem," *Mathematical Methods of Operations Research*, vol. 52, pp. 413–439, 2000.

[45] J.H. Patterson, R. Sowinski, F.B. Talbot, and J. Weglarz, "An algorithm for a general class of precedence and resource constrained scheduling problems," in *Advances in Project Scheduling*, R. Sowinski, and J. Weglarz, Amsterdam: Elsevier, 1989, pp. 3-28.

[46] J.P. Stinson, E.W. Davis, and B.M. Khumawala, "Multiple resource-constrained scheduling using branch and bound," *AIIE Transactions*, vol. 10, pp. 252-259, 1978.

[47] G. Igelmund, and F.J. Radermacher, "Preselectivestrategies for the optimization of stochastic project networks under resource constraints," *Networks*, vol. 13, pp. 1-28, 1983.

[48] A. Lova, A., P. Tormos, and F. Barber, "Multimode resource-constrained project scheduling: scheduling schemes, priority rules and mode selection rules," *Inteligenica Artificial*, vol. 10(30), pp. 69-86, 2006.

[49] E. Demeulemeester, and W. Herroelen, *Project Scheduling, A research handbook*. Boston: Kluwer Academic Publishers, 2002.

[50] R. Kolisch, A. Sprecher, "PSPLIB – A project scheduling problem library," *European Journal of Operational Research*, vol. 96, pp. 205–216, 1996.