Yugoslav Journal of Operations Research
29 (2019), Number 1, 31–42
DOI: https://doi.org/10.2298/YJOR171115025G

ON SOME IMPLEMENTATIONS OF SOLVING THE RESOURCE CONSTRAINED PROJECT SCHEDULING PROBLEMS

E.Kh. GIMADI
Sobolev Institute of Mathematics, and Novosibirsk State University, Novosibirsk, Russia
gimadi@math.nsc.ru

E.N. GONCHAROV
Sobolev Institute of Mathematics, and Novosibirsk State University, Novosibirsk, Russia
gon@math.nsc.ru

D.V. MISHIN
Sobolev Institute of Mathematics, Novosibirsk, Russia
mishindmv@gmail.com

Received: November 2017 / Accepted: September 2018

Abstract: We consider a resource-constrained project scheduling problem with respect to the makespan minimization criterion. The problem accounts for technological constraints of activities precedence together with resource constraints. Activities preemptions are not allowed. The problem with renewable resources is NP-hard in the strong sense. We propose an exact branch and bound algorithm for solving the problem with renewable resources. It uses our new branching scheme based on the representation of a schedule in form of the activity list. We use two approaches of constructing the lower bound. We present results of numerical experiments, illustrating the quality of the proposed lower bounds. The test instances are taken from the library of test instances PSPLIB.

Keywords: Project Management, Resource Constrained Project Scheduling Problem, Renewable Resources, Cumulative Resources, Branch and Bound Algorithms, PCPLIB.

MSC: 90B35, 90C27, 90C59.

1. INTRODUCTION

We consider a resource constrained project scheduling problem (RCPSP) with precedence and resource constraints. The RCPSP can be defined as a combinatorial optimization problem. A partial order on the set of activities is defined with a directed acyclic graph. For every activity, we know its duration, the list of resources required for its completion, and the amount of consumed resources. The resources are assumed to be unbounded outside the project horizon $\hat{T}$. Activities preemptions are not allowed. The objective is to schedule the activities of a project so as to minimize the project makespan.

According to the classification scheme proposed in [11], this problem is denoted as $m, 1|cpm|C_{\max}$. As a generalization of the job-shop scheduling problem the RCPSP is NP-hard in the strong sense [2] and is actually one of the most intractable classical problems in practice. It is worth noting that introducing cumulative resources into this problem, makes it solvable with polynomial complexity [8]. A remarkable example of a practically applied algorithm for solving this problem cumulative constraints type problem is the construction of the Baikal-Amur railroad in the Soviet Union in the 70-80s of the last century.

There are three main approaches for solving the RCPSP problem with renewable resources: the exact methods; heuristic methods, based on the rule of preference; and metaheuristics procedures. The branch and bound technique is the most common way to deal with this problem in order to get optimal exact solutions, i.e., it is a prominent representative of the exact methods. Some branch and bound algorithms can be found in Stinson et al. [17], Demeulemeester and Herroelen [6]. Mingozzi et al. [14] obtained several new lower bounds and presented a branch and bound scheme using lists of feasible subsets to represent partial schedules. The branch-and-bound algorithm developed by Brucker et al. [4] generalizes branch-and-bound methods for the job shop scheduling problem and the multiprocessor task scheduling problem. Among other works, we distinguish the papers of Brucker et al. [3], Sprecher [16], Baptiste and Demassey [1], Chen and Zhou [5].

In this paper we present a general definition of the problem with both renewable and cumulative constraints, as well as with restrictions on deadlines for completion of activities. The RCPSP with only cumulative resources was investigated in [8], and polynomial solvability of this problem was proved. Moreover, the authors focused on the problem with renewable resources and obtained one of the lower bounds for this problem using the algorithm for RCPSP with cumulative resources. Here we give short review of this result (see Section 3).

For the RCPSP with renewable resources, we propose a branch-and-bound algorithm with the new branching scheme, based on the presentation of a schedule in the form of an activity list. An important role in the proposed algorithm plays the construction of effective lower and upper bounds for non-truncated solutions, where lower bound is more important. In most of the cases lower bounds for the optimal solution can be calculated by relaxing some of the constraints and obtaining an exact solution for the relaxed problem. We use two approaches to

construct lower bounds. The first approach is to find the lower bound by relaxing the resource renewability condition to resource cumulativeness. After that, we solve the resulting relaxed problem and get an exact solution using the polynomial algorithm from [8]. The second approach is to use a relaxation of the original problem, where each activity is replaced by a chain of activities of unit duration. Note that the number of activities in such chain is equal to the duration of the initial activity (all activities have integer duration). Resources remain renewable. To solve this relaxed problem, we apply the algorithm [15].

The proposed lower bound algorithms were tested on instances from the electronic library PCPLIB. Both methods calculating for lower bound are compared, and sets of instances, out of series of instances of J60 with sixty activities, are revealed on which the proposed algorithms show their strength.

Another approach to solve the problem is to use metaheuristics. E.Goncharov proposed a genetic algorithm with two versions of the crossover [10]. Each crossover creates an offspring according to the criterion of optimisation of available resources. Both crossovers use heuristic rules to find promising segments of parent chromosomes for their subsequent use in the offspring. This rule is based on finding the scarcity of the resources, which, in turn, we can obtain from solutions of the relaxed problem while replacing the renewability feature with the cumulativity feature for constrained resources. To solve the relaxed problem, we use the known fast approximate algorithm [7]. Competitiveness of this genetic algorithm was demonstrated by our numerical experiments. We have found the best solutions for 9 instances from the dataset j120, and the best average deviation from the critical path lower bound for the datasets j60 (50000 and 500000 iterations) and j120 (500000 iterations).

In addition, we developed two other metaheuristics: local search algorithm and hybrid algorithm. The hybrid algorithm uses the branch and bound scheme presented in this article, as well as some elements of the genetic algorithm. The abovementioned algorithms allowed authors to find the best (previously unknown) solutions for 15 instances (at the time of writing of this paper) with 120 activities from the PSPLIB library. These results will be printed separately.

The remainder of the paper is organized as follows: Section 2 describes a general problem setting for the RCPSP. A description of the subproblem with cumulative resources is given in Section 3. Section 4 describes the subproblem with renewable resources. Section 5 contains the detailed and comparative performance tests on the benchmark datasets. The conclusions of this study are given in Section 6.

2. PROBLEM SETTING

The RCPSP problem can be defined as follows. A project is taken as a directed acyclic graph $G = (N, A)$. We denote by $N = \{1, ..., n\} \cup \{0, n + 1\}$ the set of activities in the project where activities 0 and $n+1$ are dummy. The latter activities define the start and the completion of the project respectively. The precedence relation on the set N is defined with a set of pairs $A = \{(i, j) \mid i$ – precedes $j\}$. If

$(i,j) \in A$, then activity j cannot start before activity i has been completed. The set A contains all pairs $(0,j)$ and $(j,n+1)$, $j = 1,...,n$.

The activities use renewable and cumulative (or consume nonrenewable) resources. The sets of renewable and cumulative resources are denoted by $\mathcal{K}^\rho$ and $\mathcal{K}^\nu$, respectively. For each renewable resource $k \in \mathcal{K}^\rho$, $R_k^\rho(t)$, units are available at time t. For each cumulative resource $k \in \mathcal{K}^\nu$, $R_k^\nu(t)$, units of resource of type k are arrived at time t and can be consumed at any time $t_1 \geq t$. An arbitrary activity j has deterministic duration $p_j \in Z^+$ and requires $r_{jk}(\tau) \geqslant 0$ units of resource of type k, $k \in \mathcal{K}^\rho \bigcup \mathcal{K}^\nu$ at time $\tau = 1,...,p_j$. We assume that $r_{jk}(t) \leqslant R_k^\rho(t)$, $j \in N$, $k \in \mathcal{K}^\rho$, $t = 1,...,\hat{T}$. The duration of dummy activities 0 and $n+1$ is zero, while other activities (we call them real) have nonzero durations. Moreover, dummy activities have zero resource consumption. Deadlines $d_j \in \mathbf{Z}^+$ are specified for all $j \in N_{\text{dir}} \subset N$.

Now, we introduce the problem variables. We denote by $s_j \geqslant 0$ the starting time of activity $j \in N$. Since activities are executed without preemptions, the completion time of activity j is equal to $c_j = s_j + p_j$. We define a schedule S as an $(n+2)$-vector $(s_0,...,s_{n+1})$. The completion time $T(S)$ of the project corresponds to the moment when the last activity $n+1$ is completed, i.e., $T(S) = c_{n+1}$. We denote by $J(t) = \{j \in N \mid s_j < t \leqslant c_j\}$ the set of activities which are executed in the unit time interval $[t-1,t)$ under schedule S. The problem is to find a feasible schedule $S = \{s_j\}$ respecting the resource, precedence, and deadlines constraints so that the completion time of the project is minimized. It can be formalized as follows: minimize the makespan of the project

$$C_{\max}(S) = \max_{j \in N}(s_j + p_j) \tag{1}$$

under constraints

$$s_j + p_j \leq d_j, \quad j \in N_{\text{dir}}; \tag{2}$$

$$s_i + p_i \leq s_j, \quad \forall (i,j) \in A; \tag{3}$$

$$\sum_{j \in A(t)} r_{jk}(t - s_j) \leq R_k^\rho(t), k \in \mathcal{K}^\rho, \ t \in \mathbf{Z}^+ \setminus \{0\}; \tag{4}$$

$$\sum_{t'=1}^{t} \sum_{j \in J(t')} r_{jk}(t' - s_j) \leq \sum_{t'=1}^{t} R_k^\nu(t'), \quad k \in \mathcal{K}^\nu, \ t \in \mathbf{Z}^+ \setminus \{0\}; \tag{5}$$

$$s_j \in \mathbf{Z}^+, \quad j \in N. \tag{6}$$

The set of constraints (2) guarantees compliance with the deadlines. Inequalities (3) define activities precedence constraints. Relations (4) – (5) ensure compliance with constraints on renewable and cumulative resources, respectively: the total amount of resource of type k consumed by all activities which are executed during the unit time interval [t - 1,t) must not exceed the amount of resource available at that interval. Finally, (6) defines the variables in question.

The problem (1)–(6) is known to be NP-hard if $\mathcal{K}^\rho \neq \varnothing$.

3. PROBLEM WITH CUMULATIVE RESOURCES

We consider a specific case of the problem (1)–(6) where $\mathcal{K}^\rho = \varnothing$ [8]. This case is denoted by PS^ν.

Let T be a positive integer. The solution $S^T = \{s_j^T \mid j \in N\}$ is called a *T-late schedule* if all s_j^T are maximised under constraints (2), (3), (6) and

$$s_j^T + p_j \le T, \quad j \in N.$$

The following notations are used: $Q_k(t)$ is the total amount of resource $k \in \mathcal{K}^\nu$ available at time t; $R_k^T(t)$ is the total amount of resource $k \in \mathcal{K}^\nu$ required by time t in schedule S^T; $\Delta(T) = \min\{\Delta \in \mathbf{Z}^+ \mid R_k^T(t) \le Q_k(t+\Delta);\ t = 1, \dots, T;\ k \in \mathcal{K}^\nu\}$.

It is clear that *T-late schedule S^T is feasible respecting constraints (5) if and only if* $\Delta(T) = 0$.

We put $\widetilde{T} = T_{\rm cr} + D$, where
$T_{\rm cr}$ is the critical time of the project (calculated without resource constraints);
D is the maximum deadline (later on, $\widehat{D}$ denotes the length of the binary encoding of D);
$\Delta_{\min} = \min\{\Delta \in \mathbf{Z}^+ \mid R_k^{\widetilde{T}}(t) \le Q_k(t+\Delta);\ t = 1, \dots, D;\ k \in \mathcal{K}^\nu\}$.
It is clearly, that $\Delta_{\min} \le \widetilde{\Delta} = \Delta(\widetilde{T})$.

We denote by $S_{\rm opt}$ the optimal schedule of PS^ν.

Theorem 1. *A consideration of PS^ν can be split into three special cases.*

1. If $\Delta_{\min} > 0$ or $\widetilde{\Delta} = \infty$, there is no solution for PS^ν.

2. If $\Delta_{\min} = 0$ and $0 < \widetilde{\Delta} < \infty$, the optimal schedule $S_{\rm opt} = \{s_j \mid j \in N\}$ can be computed as follows:

$$s_j = \begin{cases} s_j^{\widetilde{T}} & , \text{ for } \quad j \in N_{\rm dir}; \\ s_j^{\widetilde{T}} + \widetilde{\Delta} & , \text{ for } \quad j \in N \setminus N_{\rm dir}, \end{cases}$$

where $N_{\rm dir}$ is the set of deadline-dependent activities.
Moreover, $C_{\max}(S_{\rm opt}) = \widetilde{T} + \widetilde{\Delta}$.

3. If $\widetilde{\Delta} = 0$, we have $T_{\rm cr} \le C_{\max}(S_{\rm opt}) \le \widetilde{T}$, and the optimal solution can be found in $O(\widehat{D})$ iterations: at each iteration, the T-late schedule is computed for some $T \in \{T_{\rm cr}, \dots, \widetilde{T}\}$, and the feasibility of the schedule is checked. In other words, the optimal solution is found by the binary search.

Theorem 2. *The optimal solution for PS^ν can be found in*

$$O(\widehat{D}(u + \widetilde{I} + I \log f|K|) + |N_{\rm dir}| \log N_d)$$

time, where u is the number of arcs of the reduction graph;
$\widetilde{I}$ is the total (over all resources $k \in \mathcal{K}^\nu$) number of constancy intervals of functions

$R_k^\nu(t)$;
I is the total (over all resources and all activities) number of constancy intervals of functions $r_{jk}(t)$;
f is the width of the partial order;
$|K|$ *is the number of constrained resources;*
$|N_{\rm dir}|$ *is the number of activities with given deadlines;*
N_d *is the number of various deadlines.*

It will be interesting to establish polynomial solvability of the problem considered under assumption that intensities $R_k^\nu(t)$ of given cumulative resources can be both positive or negative.

4. PROBLEM WITH RENEWABLE RESOURCES

Now we consider another specific case of the problem (1)–(6), where $\mathcal{K}^\nu = \varnothing$ and $N_{\rm dir} = \varnothing$. This problem is denoted by PS^ρ.

We have a set of renewable resources K, for each resource type $k \in K$ there is a constant availability $R_k \in Z^+$ throughout the project horizon $\hat{T}$. Activity j has deterministic duration $p_j \in Z^+$. The profile of resource consumption is assumed to be constant for every activity. So, activity j requires $r_{jk} \geqslant 0$ units of resource of type k, $k \in K$ at every time instant when it is processed. We assume that $r_{jk} \leqslant R_k$, $j \in N$, $k \in K$. The problem is to find a feasible schedule $S = \{s_j\}$ that minimizes the completion time of the project $T(S)$, and can be formalized as follows: minimize the makespan of the project

$$T(S) = \max_{j \in N}(s_j + p_j) \tag{7}$$

under constraints

$$s_i + p_i \leq s_j, \quad \forall (i,j) \in A; \tag{8}$$

$$\sum_{j \in J(t)} r_{jk} \leqslant R_k, \ k \in K, \ t = 1, ..., \hat{T}; \tag{9}$$

$$s_j \in \mathbf{Z}^+, \quad j \in N. \tag{10}$$

Inequalities (8) define activities precedence constraints. Relation (9) corresponds to the resource constraints. Finally, (10) defines the variables in question.

4.1. A GENERAL DESCRIPTION OF THE BRANCH-AND-BOUND ALGORITHM

We represent a feasible solution as an activity list [12]. A feasible solution is encoded by the list of activities $L = (j_0, ..., j_{n+1})$. All lists under consideration are assumed to be compatible with the precedence relations. For an arbitrary list L, the serial decoding procedure (S-SGS) calculates the active schedule $S(L)$ [12]. It is known that there is an optimal schedule among the active schedules. A schedule

is called active if the starting times of the activities are such that no activity can be started earlier than its starting time without violating precedence condition and resource constraints. The parallel decoder (P-SGS) sequentially considers increasing moments of time, and schedules a subset of the eligible activities to start at this moment for each of them.

By a partial solution we mean an ordered list L_p of p activities, $p < n$, also satisfying the pre-activity relation in this list.

An unsealed vertex ν in the branch tree Ω provides two sets: ordered set L_ν is a partial solution, and unordered set D_ν is the set of vertices from the set $N \backslash L_\nu$ whose all predecessors have been scheduled up.

While examining the vertex ν in the branch tree, we perform the following actions. We apply S-SGS decoder to the partial solution L_ν and calculate the active schedule $S(L_\nu)$, and subsequently find the earliest starting times of activities and the total completion time $T_p(S(L_\nu))$ (considering the resource constraints). Then for all activities from the set D_ν, guided by these early times of their immediate predecessors, we assign the minimum to their starting times and calculate the lower bound $LB(S_\nu)$ for the network with the set of activities $N \backslash L_\nu$.

Let T^* be the minimum of the objective function found at this moment, we call it a record value, or simply a record. If $LB(S_\nu) > T^*$, then the vertex is cut off, which means that it is removed from the branch tree Ω, and we pass to the next step of the branch and bound algorithm. Otherwise, we calculate the upper bound $UB(S_\nu)$ and if $T(S_\nu) < T^*$, then we change the record. If the record has been changed, we scan all the unsealed vertices in Ω and cut off those whose lower bound are greater than the new record. After that, we modify the branch tree Ω as follows: if the set D_ν is not empty, then one of the activities of this set is added to the end of the list L_ν, the corresponded vertex is deleted from the set D_ν, and we add to D_ν those activities from $N \backslash L_\nu$, for which all their immediate predecessors belong to the modified list L_ν. If the set D_ν is empty, then no new vertex is added to the tree. At the first step of the algorithm, we assume $L_\nu = \varnothing$ and D_ν contains all the activities from the original set of activities N that do not have predecessors.

Lower bounds. We propose two methods for calculating the lower bound. In the first, we use the solution of a relaxed problem with cumulative resources, and in the second, we use the relaxation of the original problem in which the resources remain renewable, and replace each activity of integer duration by a chain of unit duration activities.

Lower bound A (LB-A). We consider the RCPSP problem, but instead of constraint (9), we introduce another constraint, were resources are cumulative:

$$\sum_{t'=1}^{t} \sum_{j \in A(t')} r_{jk} \leq \sum_{t'=1}^{t} R_k, \quad k \in K,\ t = 1, ..., \hat{T}. \tag{11}$$

In [8], it was proved that finding an exact solution of the problem (7) – (8), (10) – (11) is possible in polynomial time, and an algorithm for solving the problem was presented (see Section 3).

Classes	**n**	$\|K\|$	**AR**	**CR(% from AR)**	**SA**	p_j
1	15	4	6–10	10–30	1–2	1-4
2	15	4	6–10	10–30	2–4	1-4
3	15	4	6–10	30–60	1–2	1-4
4	15	4	6–10	30–60	2–4	1-4
5	15	4	6–10	60–90	1–2	1-4
6	15	4	6–10	60–90	2–4	1-4

Table 1: Parameters of instances classes

Lower bound B (LB-B). We consider the problem $P|prec, p_j = 1|C_{max}$, a relaxation of (7) – (10), where all activities have unit duration. To achieve this, we divide each activity of duration p_j into a chain of p_j activities of unit duration. The resource renewability property is preserved. This problem is NP-hard in the strong sense [18]. Servakh [15] developed an algorithm for solving this problem, based on the idea of dynamic programming, which is exponential in the width of the partial order. The optimal solution for this problem can be found in

$$O(\widetilde{n}^{5/2} + 2^{|C|}|C||K| \prod_{k=1}^{|C|}(S_k + 1)$$

time, where C is the set of disjoint chains in graph G,
S_k is the total duration of chain $k = 1, ...|C|$,
$\widetilde{n} = \sum_{j=1}^{n} p_j$.

To find the upper bound, we will use the greedy algorithm [9]. Advantages of this algorithm are relatively high quality of the obtained solutions and low time complexity, $n^2 \log n$ dependency from the number of activities.

5. COMPUTATIONAL EXPERIMENTS

The quality of the proposed lower bounds were tested in a series of numerical experiments. To compare the lower bounds, a graph generator was implemented. Instances were created in the same form as in the PSPlib library [13]. We considered the following parameters as inputs for generating graphs: the number of activities (n) and resource types ($|K|$), the amount of allocated resources (AR), the amount of consumed resources (as a percentage of the amount of allocated resources, CR), the durations of activities p_j and the number of successor-activities (SA). For parameters AR, CR, SA, and p_j, we use a random value between the minimum and maximum number. The instances were divided into 6 classes. Table 1 shows the parameters of these classes.

The obtained classes were used to test the effectiveness of finding lower bounds. 1000 instances for each class were generated. For each instance, the lower bounds were calculated and the average error between them was derived. The average values of the lower bounds of algorithm A were then taken as 100% watermark, and values obtained by algorithm B — as the percentage of the first algorithm. The result of comparing the two lower bounds is shown in Table 2.

Classes	LB-A, %	LB-B, %	Average width of the partial order	Average CPU-time (s) LB-A	LB-B
1	100	100,31	4,59	0,001	4,59
2	100	100,12	4,00	0,001	4,00
3	100	114,97	4,51	0,001	4,51
4	100	112,30	3,97	0,001	3,97
5	100	122,62	4,64	0,001	4,64
6	100	123,25	3,90	0,001	3,89

Table 2: Comparing lower bounds A and B

As we can see in Table 2, for instances where each activity consumes 10-30% of the allocated resources on average, it is more efficient to use the RCPSP solution with the cumulative resources (LB-A), since the error between bounds is extremely small (classes 1-2), and this algorithm is polynomial. For instances where the activity consumes on average more than 30% of allocated resources (classes 3-7), it is recommended to use the LB-B algorithm, since it finds a more accurate solution. We note, however, that it is time consuming to use the LB-B algorithm for large dimensions due to the exponential complexity of the algorithm. Average CPU-time for the algorithm LB-A is about 0,001 second for all classes of instances, and for algorithm LB-B — a few seconds. All experiments were conducted on a PC with 3.4 GHz CPU and 8 Gb RAM under the operating system Windows 7.

The next series of numerical experiments were carried out directly on the instances from the PSPlib library. Two datasets of test instances were used: J30 and J60. Each of these sets contains 48 series of instances, 10 instances in each series, 480 instances in total. For the instances from dataset J30 exact solutions are known, they are provided in PSPlib. For each series, the average deviations from the exact solution of the lower bounds obtained by the critical path method (columns 2 and 6 of Table 3), the lower LB-A bounds (columns 3 and 7), and the average deviations from the exact solution of the upper UB bounds (columns 4 and 8) were found. The average deviation of the lower bound of LB-A from the exact solutions for all 480 instances is -7.66%, while average deviation of Tcr is -9.21%. The average deviation of the upper bound UB from the exact solutions is 3.46%. The average deviations for each of the 48 series are shown in Table 3. Worth noting is that the series of instances in PSPlib have different complexity, and the LB-A algorithm has a significant advantage over the algorithm based on the critical path method on complex series.

In Table 4 similar results are provided for the set J60 of instances with 60 activities, albeit with one difference: for these instances exact solutions are unknown, and comparisons were made with the best known solutions. The average deviation of the lower LB-A bound from the best solutions on the dataset J60 is -4.37% (-7.03% for Tcr), and the average deviation of the upper UB bound is 3.75%.

We can identify instances where lower bound LB-A is better than lower bound LB-Tcr. The average deviations of LB-A for such instances were marked in bold in Tables 3 and 4.

Instances	Dev. % LB=Tcr	Dev. % LB-A	Dev. % UB	Instances	Dev. % LB=Tcr	Dev. % LB-A	Dev. % UB
J30_1	-11,55	-11,55	2,66	J30_25	-36,69	**-25,14**	13,89
J30_2	-4,03	-4,03	0,58	J30_26	-3,43	-3,43	4,06
J30_3	-2,86	-2,86	0,82	J30_27	-0,61	-0,61	0,23
J30_4	0	0	0	J30_28	0	0	0
J30_5	-27,21	**-24,46**	10,44	J30_29	-41,56	**-22,75**	11,85
J30_6	-6,86	-6,86	5,02	J30_30	-8,88	-8,88	4,89
J30_7	-3,51	-3,51	1,59	J30_31	-2,66	-2,66	0,95
J30_8	0	0	0	J30_32	0	0	0
J30_9	-33,09	**-26,30**	10,89	J30_33	-11,97	-11,97	1,27
J30_10	-5,01	-5,01	4,00	J30_34	-4,81	-4,81	1,22
J30_11	-1,33	-1,33	1,64	J30_35	-1,92	-1,92	0,20
J30_12	0	0	0	J30_36	0	0	0
J30_13	-36,49	**-19,14**	11,25	J30_37	-29,70	**-28,57**	10,71
J30_14	-6,30	**-5,92**	6,16	J30_38	-7,18	-7,18	2,52
J30_15	-1,15	-1,15	0,52	J30_39	-4,01	-4,01	0,34
J30_16	0	0	0	J30_40	0	0	0
J30_17	-11,65	-11,49	6,41	J30_41	-37,24	**-31,69**	7,63
J30_18	-7,74	-7,74	3,72	J30_42	-6,39	-6,39	2,98
J30_19	-2,64	-2,64	1,01	J30_43	-1,71	-1,71	4,12
J30_20	0	0	0	J30_44	0	0	0
J30_21	-25,67	**-24,00**	7,67	J30_45	-36,52	**-28,46**	14,26
J30_22	-7,25	-7,25	1,09	J30_46	-7,08	-7,08	4,74
J30_23	-2,41	-2,41	2,57	J30_47	-2,82	-2,82	2,03
J30_24	0	0	0	J30_48	0	0	0

Table 3: Average deviations from the exact solution for the dataset J30

Instances	Dev. % LB=Tcr	Dev. % LB-A	Dev. % UB	Instances	Dev. % LB=Tcr	Dev. % LB-A	Dev. % UB
J60_1	-8,59	-8,59	4,21	J60_25	-31,43	**-18,86**	14,84
J60_2	-2,26	-2,26	0,93	J60_26	-2,01	-2,01	3,15
J60_3	-1,89	-1,89	0,69	J60_27	0	0	0
J60_4	0	0	0	J60_28	0	0	0
J60_5	-20,96	**-17,65**	12,97	J60_29	-39,05	**-14,40**	14,02
J60_6	-0,68	-0,68	1,83	J60_30	-3,11	-3,11	4,33
J60_7	0	0	0	J60_31	0	0	0
J60_8	0	0	0	J60_32	0	0	0
J60_9	-27,73	**-14,79**	14,45	J60_33	-9,29	-9,29	4,00
J60_10	-0,15	-0,15	0,72	J60_34	-2,18	-2,18	3,08
J60_11	0	0	0	J60_35	-1,27	-1,27	1,90
J60_12	0	0	0	J60_36	0	0	0
J60_13	-39,25	**-9,72**	12,94	J60_37	-24,69	**-21,48**	15,25
J60_14	-1,02	-1,02	2,72	J60_38	-1,66	-1,66	4,47
J60_15	0	0	0	J60_39	-0,12	-0,12	0,24
J60_16	0	0	0	J60_40	0	0	0
J60_17	-8,57	-8,57	8,65	J60_41	-34,47	**-21,05**	11,46
J60_18	-0,77	-0,77	2,43	J60_42	-2,82	-2,82	3,00
J60_19	-1,25	-1,25	0,58	J60_43	-0,13	-0,13	0
J60_20	0	0	0	J60_44	0	0	0
J60_21	-24,35	**-21,55**	13,86	J60_45	-40,52	**-15,68**	13,54
J60_22	-1,69	-1,69	3,44	J60_46	-4,96	-4,96	6,11
J60_23	-0,38	-0,38	0	J60_47	0	0	0,45
J60_24	0	0	0	J60_48	0	0	0

Table 4: Average deviations from the heuristic solution for the dataset J60

6. CONCLUSIONS and SUGGESTIONS

We considered the resource constrained project scheduling problem (RCPSP) with precedence and resource constraints. The problem accounts for technological constraints of activities precedence together with resource constraints. Activities preemptions are not allowed. We proposed the exact branch and bound algorithm with a new branching scheme based on the presentation of a schedule in the form of an activity list. We used two approaches of constructing the lower bound. The results of numerical experiments are presented. A promising direction for further research is the construction of a combined algorithm using branch and bound and heuristic methods. It will be interesting to establish polynomial solvability of the problem with cumulative resources considered under assumption that intensities $R_k^{\nu}(t)$ of given cumulative resources can be both positive or negative.

Acknowledgement: The study presented in Sections 1, 3 and 4 was supported by the Russian Foundation for Basic Research (project 16-07-00829), by the Russian Ministry of Science and Education under the 5-100 Excellence Programme and by the program of fundamental scientific researches of the SB RAS I.5.1 (project 0314-2016-0014). Section 2 is supported by Russian Science Foundation (project 16-11-10041)

The authors would like to express our sincere gratitude to the anonymous referee for helpful and detailed comments that have helped to improve the quality of the manuscript.

REFERENCES

[1] Baptiste, P., Demassey, S., "Tight LP bounds for resource constrained project scheduling", *OR Spectrum*, (26) (2) (2004) 251-262.

[2] Blażewicz, J., Lenstra, J.K., Rinnoy Kan, A.H.G., "Scheduling Subject to Resource Constraints: Classification and Complexity" *Discrete Applied Math.*, (5) (1) (1983) 11–24.

[3] Brucker, P., Drexl, A., Möhring, R., at al., "Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods", *Eur. J. Oper. Res*, (112) (1) (1999) 3–41.

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

[5] Tinggui Chen, Guanglan Zhou, "Research on Project Scheduling Problem with Resource Constraints", *Journal of Software*, (8) (8) (2013) 2058-2063.

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

[7] Gimadi, E. Kh., "On Some Mathematical Models and Methods for Planning Large-Scale Projects, Models and Optimization Methods", *in Proc. AN USSR Sib. Branch, Math. Inst., Novosibirsk: Nauka*, (10) (1988) 89-115.

[8] Gimadi, E.Kh., Zalyubovskii, V.V., Sevastyanov, S.V., "Polynomial Solvability of Scheduling Prob- lems with Storable Resources and Deadlines", *Diskret. Anal. Issled. Oper., Ser. 2*, (7) (1) (2000) 9–34.

[9] Goncharov, E.N., " Stochastic Greedy Algorithm for the Resource-Constrained Project Scheduling Problem", *Diskret. Anal. Issled. Oper.*, (21) (3) (2014) 10–23.

[10] Goncharov, E.N., Leonov, V. V., "Genetic Algorithm for the Resource-Constrained Project Scheduling Problem", *Automation and Remote Control*, (78) (6) (2017) 1101-1114.

[11] Herroelen, W., Demeulemeester, E., De Reyck, B., "A Classification Scheme for Project Scheduling" in: J. Weglarz (?d.), *Project Scheduling-Recent Models, Algorithms and Applications, International Series in Operations Research and Management Science*, Kluwer Acad. Publish., Dordrecht, (14) (Chapter 1) 1998, 77–106.

[12] Kolisch, R., Hartmann, S., "Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis", in: Weglarz, (ed.) *J Project scheduling: Recent models, algorithms and applications. Kluwer Acad. Publish.*, (1999) 147–178.

[13] Kolisch, R., Sprecher, A., " PSPLIB-a Project Scheduling Problem Library", *Eur. J. Oper. Res.*, (96) (1997) 205–216. (downloadable from http://www.om-db.wi.tum.de/psplib/)

[14] Mingozzi, A., Maniezzo, V.,Ricciardelli, S., Bianco, L., "An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation", *Management Science*, 44 (1998) 715–729.

[15] Servah, V.V., "An effectively solvable case of a scheduling problem with renewable resources", *Diskret. Anal. Issled. Oper., Ser. 2*, (7) (1) (2000) 75–82.

[16] Sprecher, A., "Scheduling Resource-Constrained Projects Competitively at Modest Resource Requirements", *Management Sci.*, (46) (2000) 710–723.

[17] Stinson, J.P., Davis, E.W., Khumawala, B.M., "Multiple Resource-Constrained Scheduling Using Branch and Bound", *AIIE Transactions*, (10) (1978) 252-259.

[18] Ullman, J.D., "NP-complete scheduling problems", *J. Comput. System Sci.*, (10) (3) (1975) 284–393.