

Approximability results for the resource-constrained project scheduling problem with a single type of resources

Evgeny R. Gafarov · Alexander A. Lazarev ·
Frank Werner

© Springer Science+Business Media, LLC 2012

Abstract In this paper, we consider the well-known resource-constrained project scheduling problem. We give some arguments that already a special case of this problem with a single type of resources is not approximable in polynomial time with an approximation ratio bounded by a constant. We prove that there exist instances for which the optimal makespan values for the non-preemptive and the preemptive problems have a ratio of $O(\log n)$, where n is the number of jobs. This means that there exist instances for which the lower bound of Mingozzi et al. has a bad relative error of $O(\log n)$, and the calculation of this bound is an NP-hard problem. In addition, we give a proof that there exists a type of instances for which known approximation algorithms with polynomial time complexity have an approximation ratio of at least equal to $O(\sqrt{n})$, and known lower bounds have a relative error of at least equal to $O(\log n)$. This type of instances corresponds to the single machine parallel-batch scheduling problem $1|p\text{-batch}, b = \infty|C_{\max}$.

Keywords Project scheduling · Makespan · Lower bounds · Upper bounds

1 Introduction

Problem RCPSP may be formulated as follows. A set $N = \{1, 2, \dots, n\}$ of jobs is given. A constant amount (quantity) of $Q_k > 0$ units of resource k , $k = 1, 2, \dots, K$, is available at

E.R. Gafarov · A.A. Lazarev
Institute of Control Sciences of the Russian Academy of Sciences, Profsoyuznaya st. 65,
117997 Moscow, Russia

E.R. Gafarov
e-mail: axel73@mail.ru

A.A. Lazarev
e-mail: jobmath@mail.ru

F. Werner (✉)
Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, PSF 4120, 39016 Magdeburg,
Germany
e-mail: frank.werner@mathematik.uni-magdeburg.de

any time. Job $j \in N$ has to be processed for $p_j \geq 0$ time units without preemption. During this period, a constant amount (quantity) of $q_{jk} \geq 0$ units of resource k is used. Furthermore, finish-start precedence relations $i \rightarrow j$ are defined between the jobs according to an acyclic directed graph G . The objective is to determine the starting time S_j for each job j , $j = 1, 2, \dots, n$, in such a way that:

- at each time t , the total resource demand is less than or equal to the resource availability for each resource type;
- the given precedence relations are fulfilled and
- the makespan $C_{\max} = \max_{j=1}^n C_j$, where $C_j = S_j + p_j$, is minimized.

We use the same notations as in Brucker and Knust (2006), except the notations Q_k and q_{jk} . In that book, the authors use the symbols R_k and r_{jk} . However, usually the symbol r is used to denote the release time of a job and in order to avoid any confusion, we will use the symbols Q and q .

There is a theoretical evidence for the claim that resource-constrained project scheduling problems belong to the most intractable problems in operations research. It has a variety of applications in manufacturing, production planning, and project management. Therefore, it has become a popular playground for the latest optimization techniques. To find good solutions, various classes of algorithmic approaches have been developed.

Surveys about new results for the RCPSP are published periodically (see e.g. Kolisch and Padman 2001). There exists a special electronic library PSPLIB (Kolisch and Sprecher 1996) of experimental data to test solution algorithms for this problem. One of the best exact solution algorithms is a branch-and-bound algorithm by Brucker et al. (1998). An experimental analysis of some heuristics has been done in Hartmann and Kolisch (2000), Kolisch and Hartmann (1998, 2006). A solution algorithm for the special case with preemptions has been proposed in Demeulemeester and Herroelen (1996). It is known that in the general case, RCPSP is not in APX, where APX is the class of optimization problems that allow polynomial-time approximation algorithms with an approximation ratio bounded by a constant. This can be proved by a reduction from the graph coloring problem to a special case of the RCPSP with a large number of types of resources (Uetz 2002) (in this paper, ‘number of types of resources’ and ‘number of resources’ means the same). However, this special case is rather specific and does not correspond to practical situations.

The branch-and-bound algorithm by Brucker et al. (1998) can efficiently solve the majority of instances with up to 60 jobs. However, real-world projects contain hundreds of jobs. This algorithm, which is one of the best known exact solution algorithms, uses the lower bound of Mingozzi et al. (1998) (see Sect. 4 of this paper). Moreover, in Brucker and Knust (2006), the authors propose some dispatching rules to compute upper bounds. Thus, in this paper, we use the upper and lower bounds from Brucker and Knust (2006) to support our conjecture that the RCPSP with a single resource is not in APX.

Let C_{\max}^* be the optimal value of the objective function for the problem when preemptions are not allowed and $C_{\max}^*(pmtn)$ be the optimal value when preemptions are allowed. Without loss of generality we consider an augmented RCPSP with two dummy jobs 0 and $n + 1$, where $p_0 = p_{n+1} = 0$, $q_{0k} = q_{n+1,k} = 0$ for each resource $k = 1, 2, \dots, K$, and there are precedence relations between each triplet of jobs $0 \rightarrow j \rightarrow n + 1$, $j = 1, 2, \dots, n$. Let UB be an upper bound on the optimal makespan value C_{\max}^* , e.g., $UB = \sum_{i=1}^n p_i$. Define the *time window* $[r_i, d_i]$ for each job $i \in N$ as follows:

$$r_0 = 0, \quad r_i = \max_{j|(j \rightarrow i) \in G} \{r_j + p_j\}, \quad i = 1, 2, \dots, n + 1;$$

$$d_{n+1} = UB, \quad d_i = \min_{j|(i \rightarrow j) \in G} \{d_j - p_j\}, \quad i = n, n-1, \dots, 0.$$

According to the well-known Serial Schedule Generation Scheme (SSGS), the starting times of the jobs are determined by the following algorithm.

Algorithm LS (List Scheduling)

1. Let EL be the set of all jobs without predecessors and $Q_k(\tau) := Q_k$ for all τ , $k = 1, 2, \dots, K$.
2. If $EL = \emptyset$, then goto 10;
3. Choose one job $j \in EL$;
4. If $j = 0$, then $t := 0$ else $t := \max_{i|i \rightarrow j} \{S_i + p_i\}$;
5. If there exists a resource k for which $q_{jk} > Q_k(\tau)$ for some $\tau \in [t, t + p_j)$, then calculate the minimal value $t_k > t$ such that job j may be processed in the interval $[t_k, t_k + p_j)$ if we consider only resource k else goto 7;
6. $t := t_k$ and goto 5;
7. Schedule job j in the interval $[S_j, C_j) = [t, t + p_j)$;
8. Update the current resource profiles by setting $Q_k(\tau) := Q_k(\tau) - q_{jk}$ for all $k = 1, 2, \dots, K$ and $\tau \in [t, t + p_j)$;
9. $EL := EL \setminus \{j\}$. Add to EL all successors $i \notin EL$ of job j for which all predecessors have been scheduled and goto 2;
10. STOP.

If for a job j , the processing time p_j is equal to 0, we consider the intervals $[t, t]$ and $[S_j, C_j] = [t, t]$ under the assumption that $q_{jk} = 0$ for $k = 1, 2, \dots, K$.

There exists a sequence of choices of jobs $j \in EL$ in Step 3 which leads to an optimal schedule. We consider different dispatching rules (priority-based heuristics) to choose a job which are given in Table 1. In the last column, we give the approximation ratios for the particular rules which are proven in this paper.

In the rules EST, ECT, LST and LCT, r_i denotes the earliest starting time of job i , $r_i + p_i$ is the earliest completion time, d_i denotes the latest completion time, and $d_i - p_i$ is the latest starting time, where r_i and d_i are computed by the algorithm described above.

DEST is the dynamic version of the EST rule, i.e., the values r_i are computed after each step of Algorithm LS. Here r_i is the earliest starting time at which one can begin to process job i , if we take into account the jobs already scheduled. DECT is the dynamic version of the ECT rule, i.e., the values d_i are computed after each step of Algorithm LS. Here d_i is the earliest completion time of job i , if we take into account the jobs already scheduled.

In this paper, we consider only the special case of the problem with a single resource 1 and an amount of the resource Q_1 . Sometimes, we will also use the notations $q_i = q_{i1}$ and $Q = Q_1$. Even for this case, we get only *negative* approximability results.

The rest of this paper is organized as follows. In Sect. 2, we analyze upper and lower bounds for special cases of the RCPSP. Then, in Sects. 3 and 4, upper and lower bounds for the general case are investigated. In Sect. 5, we give some remarks about the approximability of the RCPSP. An NP-hardness proof for a badly approximable special case of the RCPSP is presented in Sect. 6.

2 Upper and lower bounds for special cases

In this section, we consider three special cases of the RCPSP with a single type of resources (i.e., with a single resource).

Table 1 Dispatching rules (priority-based heuristics)

Notation	Description	Approxim. ratio
job-based		
SPT	choose a job with the smallest processing time	$O(n)$
LPT	choose a job with the largest processing time	$O(n)$
network-based		
MIS	choose a job with the most immediate successors	$O(\sqrt{n})$
LIS	choose a job with the least immediate successors	$O(\sqrt{n})$
MTS	choose a job with the most total successors	$O(\sqrt{n})$
LTS	choose a job with the least total successors	$O(\sqrt{n})$
GRPW	choose a job with the greatest rank positional weight, i.e., with the largest total processing time of all successors	$O(\sqrt{n})$
critical path-based		
EST	choose a job with the smallest earliest starting time	$O(n)$
ECT	choose a job with the smallest earliest completion time	$O(n)$
LST	choose a job with the smallest latest starting time	$O(n)$
LCT	choose a job with the smallest latest completion time	$O(n)$
MSLK	choose a job with the smallest slack $d_i - r_i - p_i$,	$O(n)$
MW	choose a job with the smallest window $d_i - r_i$,	$O(n)$
DEST (Dynamic EST)	choose a job with the smallest earliest starting time	$O(n)$
DECT (Dynamic ECT)	choose a job with the smallest earliest completion time	$O(\sqrt{n})$
resource-based		
GRR	choose a job with the greatest resource requirements	$O(n)$

Strip-packing problem. Given a horizontal strip of height Q_1 , and a list L of rectangular pieces $\{1, 2, \dots, n\}$, pack the pieces into the strip such that the width SPP to which the strip is filled is as small as possible. The pieces are not allowed to overlap. We also assume that each piece $i \in L$ is defined by its width p_i and height q_{i1} .

For the special case **PMS (Parallel machine scheduling)**, we have $q_i = 1$ for $i = 1, 2, \dots, n$ and $Q_1 = m \in \mathbb{Z}$, where m is the number of machines.

For the special case **LSPP (like a Strip packing problem)**, we have an empty graph of precedence relations. This special case of the RCPS is similar to the strip packing problem.

For the special case **UPT (Unit processing time)**, we have $p_i = 1$ for $i = 1, 2, \dots, n$.

All three special cases are NP-hard in the strong sense and in APX. Moreover, for each of them, there exist lower bounds with a relative error bounded by a constant, which are summarized in Table 2. Upper bounds for special cases are obtained by Algorithm LS. In this table, CP denotes the length of a critical path in the precedence graph G . We note that for the LSPP, it has been shown in Lazarev and Gafarov (2008), Gafarov et al. (2010) that

$$C_{\max}^* \leq C_{\max}(LS_{DEST}) \leq 2 \frac{\sum_{i=1}^n p_i q_i}{Q_1} + p_{\max},$$

where $\frac{\sum_{i=1}^n p_i q_i}{Q_1}$ and p_{\max} are lower bounds for C_{\max}^* , and $C_{\max}(LS_{DEST})$ is the objective function value of a schedule obtained by Algorithm LS according to dispatching rule $DEST$. It is known (Rykov 2006) that, if we consider instances with the same parameters, for two

Table 2 Lower and upper bounds for special cases

Special case	Upper bound	Lower bound	NP-hardness. Reduction from
PMS	Dispatching rule: any	$C_{\max}^* < \frac{\sum_{i=1}^n p_i}{m} + CP$.	Clique (if $p_j = 1$)
	Approx. ratio is less than 2 (Lawler et al. 1989)	Relative error of $LB = \max\{\frac{\sum_{i=1}^n p_i}{m}, CP\}$ is less than 2 (Lawler et al. 1989)	Partition (if $m = 2$)
LSPP	Dispatching rule: DEST	$C_{\max}^* < 2 \cdot LB =$ $2 \max\{\frac{\sum_{i=1}^n p_i q_i}{Q_1}, p_{\max}\}$.	Bin-packing (if $p_j = 1$)
	Approx. ratio is less than 3 (Lazarev and Gafarov 2008; Gafarov et al. 2010)	Relative error of LB is less than 2 (Lazarev and Gafarov 2008; Gafarov et al. 2010)	Partition (if $q_j = 1$)
UPT	Dispatching rule: EST	$C_{\max}^* < 2(\frac{\sum_{i=1}^n q_i}{Q_1} + CP)$.	Bin-packing
	Approx. ratio is less than 4 (Gafarov et al. 2010) (Sect. 2)	Relative error of $LB = \max\{\frac{\sum_{i=1}^n q_i}{Q_1}, CP\}$ is less than 4 (Gafarov et al. 2010) (Sect. 3)	Clique (if $q_j = 1$)

optimal solutions of the LSPP and the SPP (Strip Packing Problem) we have the relation

$$1 \leq \frac{SPP^*}{LSPP^*} \leq 2.7,$$

where $LSPP^*$ and SPP^* are the optimal objective function values. For the Strip packing problem, it is known that

$$SPP^* \leq 2 \max\left\{\frac{\sum_{i=1}^n q_i \cdot p_i}{Q_1}, p_{\max}\right\}.$$

The proof was given in Martello et al. (2003) (which is based on a result from Steinberg (1997)). For UPT, approximability results with detailed but rather long proofs have been presented in Gafarov et al. (2010).

3 Upper bounds for the general case

In this section, we consider classical dispatching rules to select a job in Step 3 of Algorithm LS. We use the same list of rules as in Brucker and Knust (2006) (see Table 1, where the rules DEST and DECT are mentioned in addition).

We denote by $C_{\max}(LS_a)$ the objective function value of a schedule obtained by Algorithm LS according to dispatching rule a . Here we show that for each of these dispatching rules, there exists an instance for which we have

$$\frac{C_{\max}(LS_a)}{C_{\max}^*} = O(n) \quad (\text{or } O(\sqrt{n})).$$

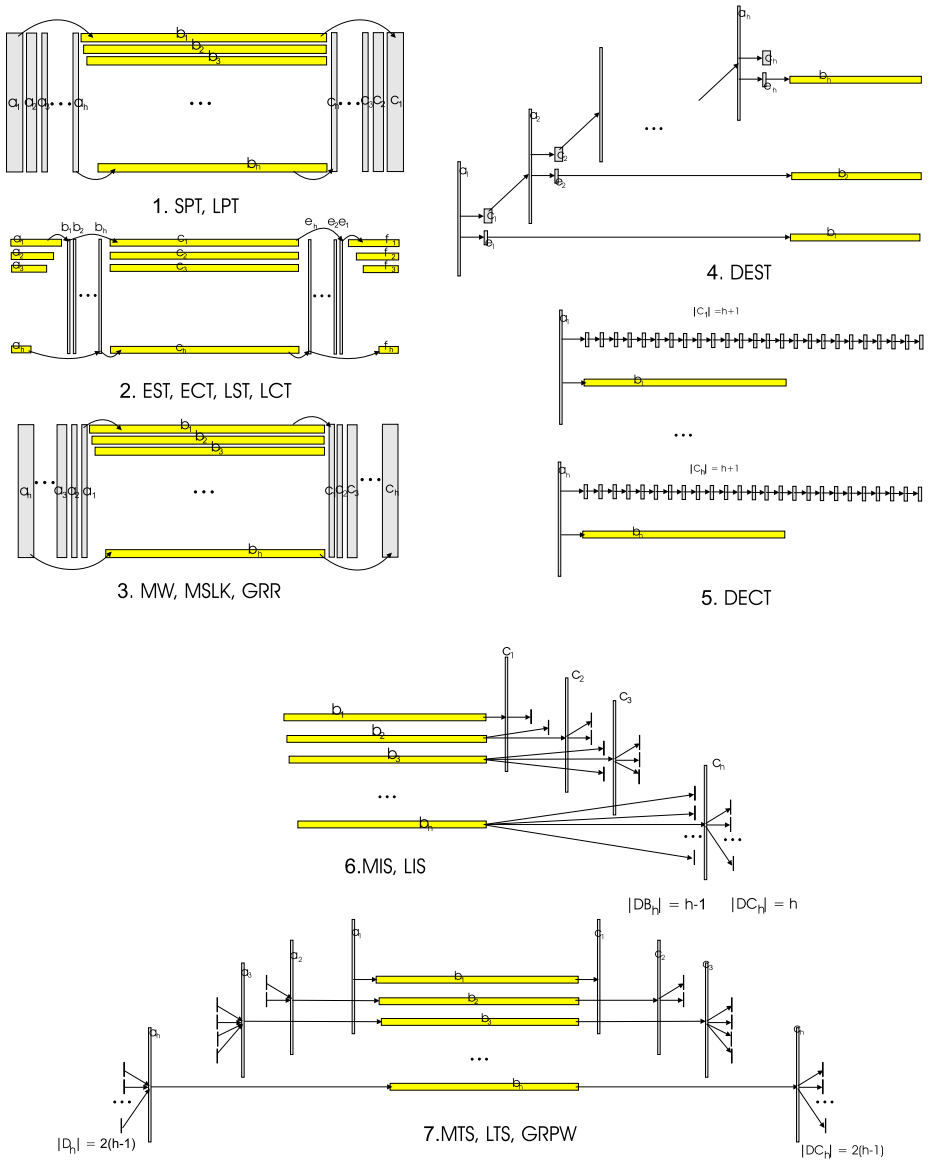


Fig. 1 Dispatching rules

Lemma 1 *There exists an instance of the RCPSP for which*

$$\frac{C_{\max}(LS_a)}{C_{\max}^*} = O(n), \quad a \in \{SPT, LPT\}.$$

Proof We consider the instance from Fig. 1 (1. SPT, LPT). We have h chains of jobs $a_i \rightarrow b_i \rightarrow c_i$, $i = 1, 2, \dots, h$. For each job b_i , $i = 1, 2, \dots, h$, we have $p_{b_i} = p - \varepsilon(i - 1)$, $q_{b_i} = 1$. For each job a_i , $i = 1, 2, \dots, h$, we have $p_{a_i} = \varepsilon(h - i + 1)$, $q_{a_i} = h$. For each job

$c_i, i = 1, 2, \dots, h$, we have $p_{c_i} = \varepsilon(h - i + 1), q_{c_i} = h$, where $h^2\varepsilon \ll p$ (here and in the following, we assume $\varepsilon = 1/(n \cdot p)^n, p \in Z_+$) and $Q_1 = h$.

For this instance, an optimal schedule corresponds to a sequence

$$\pi = (a_1, a_2, \dots, a_h, b_1, b_2, \dots, b_h, c_1, c_2, \dots, c_h)$$

of job choices but for the dispatching rule SPT, we get the sequence

$$\pi_{SPT} = (a_h, a_{h-1}, \dots, a_1, b_h, c_h, b_{h-1}, c_{h-1}, \dots, b_1, c_1)$$

of choices, i.e., all jobs b_1, b_2, \dots, b_h are processed successively without overlapping. Thus, we get $C_{\max}^* = 2(h + (h - 1) + \dots + 1)\varepsilon + p$ and $C_{\max}(LS_{SPT}) = 2(h + (h - 1) + \dots + 1)\varepsilon + p + (p - \varepsilon) + (p - 2\varepsilon) + \dots + (p - (h - 1)\varepsilon)$. Therefore, we obtain

$$\lim_{\varepsilon \rightarrow 0} \frac{C_{\max}(LS_{SPT})}{C_{\max}^*} = h = \frac{n}{3}.$$

In an analogous way, we may consider the sequence

$$\pi_{LPT} = (a_1, b_1, c_1, a_2, b_2, c_2, \dots, a_h, b_h, c_h)$$

of job choices. □

We denote as the *reverse instance* an instance in which the orientation of all arcs in the precedence graph is opposite but the remaining data of the instance are the same. We have to note that the instance from Lemma 1 is also *bad* for the reverse instance, i.e., if we use this dispatching rule for the reverse instance, then the approximation ratio remains the same. All other instances presented in this paper have this property, too.

The proof of the remaining upper bounds is similar for each of the dispatching rules mentioned above. We prove that for each rule a in a corresponding schedule π_a for a particular instance, all ‘long’ jobs are processed successively without overlapping and as a consequence, the approximation ratio is approximately equal to $O(n)$ (or $O(\sqrt{n})$). Thus, for the following lemmas, we present short proofs in which only the parameters of the jobs, an optimal sequence π^* and the sequence π_a are described. The complete proofs (which are rather technical and require 1.5 pages more) have been given in Gafarov et al. (2010).

Lemma 2 *There exists an instance of the RCPS for which*

$$\frac{C_{\max}(LS_a)}{C_{\max}^*} = O(n), \quad a \in \{EST, LST, ECT, LCT\}.$$

Proof We consider the static versions of these rules, i.e., the values r_i, d_i are computed only once before the use of Algorithm LS. Let us consider the instance given in Fig. 1 (EST, ECT, LST, LCT). We have h chains of jobs $a_i \rightarrow b_i \rightarrow c_i \rightarrow e_i \rightarrow f_i, i = 1, 2, \dots, h$. For each job $b_i, e_i, i = 1, 2, \dots, h$, we have $p_{b_i} = p_{e_i} = \varepsilon, q_{b_i} = q_{e_i} = h$. For each job $a_i, f_i, i = 1, 2, \dots, h$, we have $p_{a_i} = p_{f_i} = 2\varepsilon(h - i + 1), q_{a_i} = q_{f_i} = 1$. For each job $c_i, i = 1, 2, \dots, h$, we have $p_{c_i} = p, q_{c_i} = 1$, where $h^2\varepsilon \ll p$ and $Q_1 = h$. We get $\pi^* = (a_1, a_2, \dots, a_h, b_1, b_2, \dots, b_h, c_1, c_2, \dots, c_h, e_1, e_2, \dots, e_h, f_1, f_2, \dots, f_h), \pi_{EST} = (a_1, a_2, \dots, a_h, b_h, c_h, e_h, f_h, b_{h-1}, c_{h-1}, e_{h-1}, f_{h-1}, \dots, b_1, c_1, e_1, f_1)$ and $\pi_{LST} = (a_1, b_1, c_1, e_1, f_1, \dots, a_h, b_h, c_h, e_h, f_h)$.

For $a \in \{ECT, LCT\}$, we have $p_{c_i} = p - 4(i - 1)\varepsilon$. We get $\pi^* = (a_1, a_2, \dots, a_h, b_1, b_2, \dots, b_h, c_1, c_2, \dots, c_h, c_k, e_1, e_2, \dots, e_h, f_1, f_2, \dots, f_h)$, $\pi_{ECT} = (a_h, b_h, a_{h-1}, b_{h-1}, \dots, a_1, b_1, c_h, e_h, f_h, c_{h-1}, e_{h-1}, f_{h-1}, \dots, c_1, e_1, f_1)$ and $\pi_{LCT} = (a_1, b_1, c_1, e_1, f_1, \dots, a_h, b_h, c_h, e_h, f_h)$ \square

Lemma 3 *There exists an instance of the RCPSP for which*

$$\frac{C_{\max}(LS_a)}{C_{\max}^*} = O(n), \quad a \in \{MW, MSLK, GRR\}.$$

Proof We consider the instance from Fig. 1 (3. MW, MSLK, GRR). We have h chains of jobs $a_i \rightarrow b_i \rightarrow c_i$, $i = 1, 2, \dots, h$. For each job b_i , $i = 1, 2, \dots, h$, we have $p_{b_i} = p + (h - i)\varepsilon$, $q_{b_i} = 1 + i\varepsilon$. For each job a_i, c_i , $i = 1, 2, \dots, h$, we have $p_{a_i} = p_{c_i} = i\varepsilon$, $q_{a_i} = q_{c_i} = h + 1$, where $\varepsilon < 1/h^2$ and $Q_1 = h + 1$.

We get $\pi^* = (a_1, a_2, \dots, a_h, b_1, b_2, \dots, b_h, c_1, c_2, \dots, c_h)$, $\pi_{MW} = (a_h, a_{h-1}, \dots, a_1, b_h, c_h, b_{h-1}, c_{h-1}, \dots, b_1, c_1)$ since $d_{a_h} - r_{a_h} = \dots = d_{a_1} - r_{a_1} = UB - (p + h\varepsilon)$, $d_{b_h} - r_{b_h} = UB - 2h\varepsilon < d_{b_{h-1}} - r_{b_{h-1}} = UB - 2(h - 1)\varepsilon < \dots < d_{b_1} - r_{b_1}$ and $d_{c_h} - r_{c_h} = UB - p - h\varepsilon < d_{b_{h-1}} - r_{b_{h-1}} < \dots < d_{b_1} - r_{b_1}$, and so on. Moreover, we obtain $\pi_{MSLK} = (a_h, b_h, c_h, a_{h-1}, b_{h-1}, c_{h-1}, \dots, a_1, b_1, c_1)$, where $d_{a_i} - r_{a_i} - p_{a_i} = d_{b_i} - r_{b_i} - p_{b_i} = d_{c_i} - r_{c_i} - p_{c_i}$ for $i = 1, 2, \dots, h$ and $\pi_{GRR} = (a_1, a_2, \dots, a_h, b_h, c_h, b_{h-1}, c_{h-1}, \dots, b_1, c_1)$. \square

Lemma 4 *There exists an instance of the RCPSP for which*

$$\frac{C_{\max}(LS_{DEST})}{C_{\max}^*} = O(n).$$

Proof We consider the instance from Fig. 1 (4. DEST). We have a tree of jobs which contains subsets of jobs $\{a_i, b_i, c_i, e_i\}$, $a_i \rightarrow e_i \rightarrow b_1$, $a_i \rightarrow c_i$, $i = 1, 2, \dots, h$, $c_i \rightarrow a_{i+1}$, $i = 1, 2, \dots, h - 1$. For each job b_i , $i = 1, 2, \dots, h$, we have $p_{b_i} = p$, $q_{b_i} = 1$. For each job a_i , $i = 1, 2, \dots, h$, we have $p_{a_i} = \varepsilon$, $q_{a_i} = h$. For each job c_i , $i = 1, 2, \dots, h$, we have $p_{c_i} = 2\varepsilon$, $q_{c_i} = \varepsilon$. For each job e_i , $i = 1, 2, \dots, h$, we have $p_{e_i} = \varepsilon$, $q_{e_i} = \varepsilon$. The precedence relations are given in Fig. 1. We assume $h^2\varepsilon \ll p$ and $Q_1 = h$.

We get $\pi^* = (a_1, c_1, e_1, a_2, c_2, e_2, \dots, a_h, c_h, e_h, b_1, b_2, \dots, b_h)$ and $\pi_{DEST} = (a_1, c_1, e_1, b_1, \dots, a_h, c_h, e_h, b_h)$. \square

Lemma 5 *There exists an instance of the RCPSP for which*

$$\frac{C_{\max}(LS_{DECT})}{C_{\max}^*} = O(\sqrt{n}).$$

Proof We consider the instance from Fig. 1 (5. DECT). We have h independent sets of jobs such that the i -th set contains the set of jobs $\{a_i, b_i\} \cup C_i$, $a_i \rightarrow b_i$, $a_i \rightarrow C_i$, where $|C_i| = h + 1$, $C_i = \{j_1^i, \dots, j_{h+1}^i\}$, $i = 1, 2, \dots, h$. For each job b_i , $i = 1, 2, \dots, h$, we have $p_{b_i} = p$, $q_{b_i} = 1$. For each job a_i , $i = 1, 2, \dots, h$, we have $p_{a_i} = 2\frac{p}{h}$, $q_{a_i} = h + 1$. For each job $c \in C_i$, $i = 1, 2, \dots, h$, we have $p_c = \frac{p}{h}$, $q_c = \frac{1}{h}$. Moreover, let $Q_1 = h + 1$.

We get $\pi^* = (a_1, a_2, \dots, a_h, b_1, b_2, \dots, b_h, C_1, C_2, \dots, C_h)$ and $\pi_{DECT} = (a_1, C_1, b_1, a_2, C_2, b_2, \dots)$. \square

Lemma 6 *There exists an instance of the RCPSP for which*

$$\frac{C_{\max}(LS_a)}{C_{\max}^*} = O(\sqrt{n}), a \in \{MIS, LIS\}.$$

Proof We consider the instance from Fig. 1 (6. MIS, LIS). We have h independent sets of jobs, each of them includes a chain $b_i \rightarrow c_i$ and the subsets $DB_i, DC_i, i = 1, 2, \dots, h$, where $b_i \rightarrow DB_i$ and $c_i \rightarrow DC_i$. The notation $j \rightarrow X_i$ means that for all $l \in X_i$, we have $j \rightarrow l$. In addition, we have $|DB_i| = (i - 1), |DC_i| = i$ for $i = 1, 2, \dots, h$. For all jobs l from these subsets, we have $p_l = \varepsilon, q_l = \varepsilon$.

For each job $b_i, i = 1, 2, \dots, h$, we have $p_{b_i} = p + (h - i)\varepsilon, q_{b_i} = 1$. For each job $c_i, i = 1, 2, \dots, h$, we have $p_{c_i} = \varepsilon, q_{c_i} = h$, where $h^2\varepsilon \ll p$ and $Q_1 = h$.

We get $\pi^* = (b_1, b_2, \dots, b_h, c_1, DC_1, DB_2, DB_3, \dots, DB_h, c_2, c_3, \dots, c_h, DC_2, \dots, DC_h)$ and $\pi_{MIS} = (b_h, c_h, b_{h-1}, c_{h-1}, \dots, b_1, c_1, DB_2, \dots, DB_h, DC_1, DC_2, \dots, DC_h)$.

For rule LIS, we have $p_{b_i} = p - (h - i)\varepsilon, i = 1, 2, \dots, h$, and q_{b_i} as before. We get $\pi_{LIS} = (b_1, c_1, DC_1, b_2, DB_2, c_2, DC_2, b_3, DB_3, c_3, DC_3, \dots, b_h, DB_h, c_h, DC_h)$.

It is easy to construct an analogous instance with a symmetric graph of precedence relations (see the idea in the proof of Theorem 1 later). □

Lemma 7 *There exists an instance of the RCPSP for which*

$$\frac{C_{\max}(LS_a)}{C_{\max}^*} = O(\sqrt{n}), a \in \{MTS, LTS, GRPW\}.$$

Proof We can prove this lemma in an analogous way as Lemma 6. For an illustration, see Fig. 1 (7. MTS, LTS, GRPW). We have h independent sets of jobs, each of them includes a chain $a_i \rightarrow b_i \rightarrow c_i$ and the subsets $D_i, DC_i, i = 1, 2, \dots, h$, where $D_i \rightarrow a_i$ and $c_i \rightarrow DC_i$ and $|DC_i| = |D_i| = 2(i - 1), i = 1, 2, \dots, h$. For all jobs l from these subsets, we have $p_l = \varepsilon, q_l = \varepsilon$.

For each job $b_i, i = 1, 2, \dots, h$, we have $p_{b_i} = p + (h - i)\varepsilon, q_{b_i} = 1$. For each job $a_i, c_i, i = 1, 2, \dots, h$, we have $p_{a_i} = p_{c_i} = \varepsilon, q_{a_i} = q_{c_i} = h$, where $h^2\varepsilon \ll p$ and $Q_1 = h$.

We get $\pi^* = (D_1, \dots, D_h, a_1, \dots, a_h, b_1, \dots, b_h, c_1, \dots, c_h, DC_1, \dots, DC_h)$ and $\pi_{MTS} = (D_h, a_h, b_h, c_h, D_{h-1}, a_{h-1}, b_{h-1}, c_{h-1}, \dots, D_1, a_1, b_1, c_1, DC_1, DC_2, \dots, DC_h)$.

For rule LTS, we have $p_{b_i} = p - (h - i)\varepsilon, i = 1, 2, \dots, h$, and q_{b_i} as before. □

We can present the following result.

Theorem 1 *There exists an instance of the RCPSP for which*

$$\frac{C_{\max}(LS_a)}{C_{\max}^*} = O(\sqrt{n})$$

holds for all dispatching rules a mentioned above.

Proof We construct such an instance as follows. The instance contains the subsets of jobs N_1, N_2, \dots, N_{10} . Each subset of jobs $N_i, i = 1, 2, \dots, 10$, corresponds to the instance from Lemmas 1–7 (in Lemmas 2, 6 and 7, we have altogether six instances). In each subset $N_i, i = 1, 2, \dots, 10$, we have h jobs with long processing times approximately equal to p .

In addition, we have 9 dummy jobs o_1, o_2, \dots, o_9 such that for all $j \in N_i$ and $l \in N_{i+1}$, $i = 1, 2, \dots, 9$, we have $j \rightarrow o_i \rightarrow l$.

Then, in an optimal schedule, the long jobs from the same subset are processed in parallel. However, if we use any of the above dispatching rules for this instance, we have a subset N_i , where the long jobs are processed successively without overlapping. \square

4 Lower bounds for the general case

From Lazarev and Gafarov (2008), it is known that well-known ‘polynomial’ lower bounds on the optimal objective function value of the problem have ‘bad’ approximation ratios.

Let *CritP* (Critical Path) be a chain of jobs (vertices in the graph of precedence relations) with the maximal total processing time, i.e., the maximal sum of the processing times of the jobs from this chain. *CP* denotes the total processing time of *CritP*, and $j \in \text{CritP}$ denotes a job j from *CritP*.

Denote $LB_0 = CP$. It is evident that LB_0 is a lower bound for the optimal objective function value C_{\max}^* . Denote

$$LB_1 = \sum_{j=1}^n q_{jk} p_j / Q_k$$

and

$$LB_S = CP + \max_{j \notin \text{CritP}} \{ \max\{p_j - e_j, 0\} \},$$

where e_j is the maximum length of an interval contained in $[r_j, d_j]$, in which job j can be simultaneously processed with the activities from a critical path *CritP* without violating the resource constraints.

It is known (Lazarev and Gafarov 2008) that there exists an instance for which $\frac{C_{\max}^*}{LB_0} = n$. Moreover, there exists an instance for which $\frac{C_{\max}^*}{LB_1} = n$, and there exists an instance for which $\frac{C_{\max}^*}{LB_S} = n/2$.

In Mingozi et al. (1998), Uetz (2002), Bianco and Caramia (2011), Carlier and Neron (2003), specific algorithms have been presented for the calculation of lower bounds. One of these bounds is the lower bound given by Mingozi et al. (1998).

4.1 Lower bound given by Mingozi et al.

We consider a linear programming formulation that partially relaxes the precedence relations and allows preemption. The columns of this linear program *LP* correspond to so-called non-dominated feasible subsets. A feasible set X is a set of jobs that may be processed simultaneously, i.e., there are no precedence relations between any pair of $i, j \in X$ and all resource constraints are satisfied (i.e., $\sum_{i \in X} q_{ik} \leq Q_k$ for $k = 1, 2, \dots, K$). Such a set is called non-dominated if it is not a proper subset X of another feasible set Y . We consider all non-dominated feasible sets and in addition all one-element sets $\{i\}$ for $i = 1, 2, \dots, n$.

We denote all these sets by X_1, X_2, \dots, X_f , where f is the number of such sets, and associate with each set X_j an incidence vector $a^j \in \{0, 1\}^n$ defined by $a_i^j = 1$ if $i \in X_j$, and $a_i^j = 0$ otherwise, $j = 1, 2, \dots, f$.

Furthermore, let x_j be a variable denoting the number of time units over which all jobs in X_j are processed in parallel. Then the following linear program problem provides a lower bound LB_M for the RCPSP by relaxing the precedence relations and allowing preemption:

$$\begin{aligned} & \sum_{j=1}^f x_j \longrightarrow \min \\ \text{s.t.} & \\ & \sum_{j=1}^f a_i^j x_j \geq p_i, \quad i = 1, 2, \dots, n; \\ & x_j \geq 0, \quad j = 1, 2, \dots, f. \end{aligned}$$

It is known that the calculation of LB_M is an NP-hard problem (by a reduction from the well-known NP-hard ‘Bin packing’ problem) (Lazarev and Gafarov 2008), and there exist instances for which $\frac{C_{\max}^*}{LB_M} \approx 2$ (see Gafarov et al. 2010, Lemma 13).

In Lazarev and Gafarov (2008), there was given a conjecture that $C_{\max}^* < 2 \cdot C_{\max}^*(pmtn)$. This conjecture is true for the special cases PMS, LSPP and for a special case, for which there are only one or two preempted jobs in an optimal schedule for the RCPSP with preemptions. However, the conjecture is false for the general case.

Theorem 2 *There exists an instance of the RCPSP for which*

$$\frac{C_{\max}^*}{C_{\max}^*(pmtn)} = O(\log n).$$

Proof We consider an instance of the type given in Fig. 2(a). For this instance, we have $m + 1$ levels of jobs. At the highest level, we have a job j_1^1 with processing time Mp , at the second highest level, we have two jobs j_1^2 and j_2^2 with processing times $\frac{M}{2}p$, at level h , $h = 2, 3, \dots, m$, we have 2^{h-1} jobs j_i^h , $i = 1, 2, \dots, 2^{h-1}$, with processing time $p_{j_i^h} = \frac{M}{2^{h-1}}p$, and so on. At the lowest level, we have a chain of short and very short jobs $e_1 \rightarrow j_1^{m+1} \rightarrow e_2 \rightarrow j_2^{m+1} \rightarrow \dots \rightarrow e_M \rightarrow j_M^{m+1} \rightarrow e_{M+1}$, where $p_{e_i} = \varepsilon$, $q_{e_i} = m + 1$, $i = 1, 2, \dots, M + 1$, and $p_{j_i^{m+1}} = p$, $q_{j_i^{m+1}} = 1$, $i = 1, 2, \dots, M$. For each job j_i^h of level h , $h = 2, 3, \dots, m$, we have $p_{j_i^h} = \frac{M}{2^{h-1}}p$, $q_{j_i^h} = 1$. At each level h , $h = 2, 3, \dots, m$, we have a chain of jobs $e_1 \rightarrow j_1^h \rightarrow e_{\frac{M}{2^{h-1}}+1} \rightarrow j_2^h \rightarrow e_{2 \cdot \frac{M}{2^{h-1}}+1} \rightarrow \dots \rightarrow j_{2^{h-1}}^h \rightarrow e_{M+1}$. Some of these precedence relations are illustrated in Fig. 2(a). For this instance, we have $(M + 1)\varepsilon \ll p$ (e.g. $\varepsilon = 1/((M + 1)p)^2$, $p \in \mathbb{Z}_+$). In addition, we have $Q_1 = m + 1 = q_{e_i}$, $i = 1, 2, \dots, M + 1$, and $M = 2^m$. By dotted lines, we separate the jobs which will be processed in parallel in an optimal schedule for the non-preemptive problem.

In Fig. 2(b), for such an instance with only $m + 1 = 6$ levels, we give the resulting optimal solution. For this instance, we obtain $C_{\max}^*(pmtn) = 32p + 33\varepsilon$ and $C_{\max}^* = 112p + 33\varepsilon$.

For the general case, we have $C_{\max}^*(pmtn) = Mp + (M + 1)\varepsilon$ and $C_{\max}^* = Mp + \frac{m}{2}Mp + (M + 1)\varepsilon$. Hence, we obtain

$$\frac{C_{\max}^*}{C_{\max}^*(pmtn)} \approx \frac{m + 2}{2}.$$

from which we obtain

$$m = \log \frac{n}{3}.$$

Therefore, we get

$$\frac{C_{\max}^*}{C_{\max}^*(pmtn)} \approx \frac{m+2}{2} = \frac{\log n - \log 3 + 2}{2}. \quad \square$$

Corollary 1 Given a set $\{LB^1, LB^2, \dots, LB^l\}$ of lower bounds based on the preemption of the jobs, it follows from $LB^k \leq C_{\max}^*(pmtn)$, $k = 1, 2, \dots, l$, that there exists an instance for which $\frac{C_{\max}^*}{LB^k} = O(\log n)$, $k = 1, 2, \dots, l$.

Obviously, this also holds for the lower bound by Mingozzi et al., i.e., there exists an instance of the RCPSP for which $\frac{C_{\max}^*}{LB_M} = O(\log n)$ since $LB_M \leq C_{\max}^*(pmtn)$, and we can formulate the following result:

Theorem 3 There exists a type of instances of the RCPSP for which

$$\frac{C_{\max}^*}{LB_M} = O(\log n),$$

and the calculation of LB_M for this type is an NP-hard problem.

The idea of constructing such instances is not difficult. This instance contains two subsets of jobs N_1 and N_2 . The jobs from the first subset correspond to the instance illustrated in Fig. 2(a), where $Q = m + 1$ and $|N_1| = n$. In the set N_2 , we have n independent jobs with unit processing times $p_j = 1$ and $\sum_{j \in N_2} q_j = 2m + 2$. In addition, we have a dummy job o_1 such that $j \rightarrow o_1 \rightarrow l$ for all $j \in N_1, l \in N_2$. We can present a reduction from the partition problem to the problem of calculating LB_M for this type of instances. The partition problem is as follows. Given is a set $N = \{b_1, b_2, \dots, b_n\}$ of numbers $b_1 \geq b_2 \geq \dots \geq b_n > 0$ with $b_i \in \mathbb{Z}_+$, $i = 1, 2, \dots, n$. Does there exist a subset $N' \subset N$ such that $\sum_{i \in N'} b_i = \frac{1}{2} \sum_{i \in N} b_i$? If we have n integers b_1, b_2, \dots, b_n in the instance of the partition problem, then we assume $q_{n+j} = b_j$, $j \in N_2 = \{n + 1, n + 2, \dots, n + n\}$.

The branch-and-bound algorithm by Brucker et al. (1998), which uses LB_M as a basic lower bound, can efficiently solve the majority of benchmark instances from PSPLIB with up to 60 jobs. Brucker and Knust (see Brucker and Knust 2006, p. 125) note that the number of non-dominated sets (for the definition, see above) considered within the computation of LB_M grows exponentially for some of these instances. By Theorem 3, we try to argue concerning the practical weakness of this lower bound observed in Brucker et al. (1998), Brucker and Knust (2006).

5 Some further remarks

We can consider the optimal objective function values for the special cases *PMS*, *LSPP* and *UPT* as lower bounds for the optimal value C_{\max}^* in the general case. For example, if we delete all precedence relations from the original instance, we have an instance of the special case *LSPP*. If we split any job j with $p_j > 1$ into a set of jobs each of them with a processing

time equal to 1, we have a *UPT* instance, and so on. In Sect. 3, we have stated (see Table 2 and Sect. 4 for the definitions of LB_0 and LB_1) that

$$C_{\max}^{PMS^*} < LB_0 + LB_1 \leq 2LB_M,$$

$$C_{\max}^{LSPP^*} < 2 \max\{LB_0, LB_1\} \leq 2LB_M$$

and

$$C_{\max}^{UPT^*} < 2(LB_0 + LB_1) \leq 4LB_M$$

(e.g., $C_{\max}^{LSPP^*}$ is the optimal objective function value for the relaxed instance which corresponds to the special case LSPP). Therefore, we can prove the following theorem:

Theorem 4 *There exists an instance of the RCPSP for which*

$$\frac{C_{\max}^*}{C_{\max}^{PMS^*}} = O(\log n), \quad \frac{C_{\max}^*}{C_{\max}^{LSPP^*}} = O(\log n), \quad \frac{C_{\max}^*}{C_{\max}^{UPT^*}} = O(\log n).$$

If we do not take into consideration the non-preemptive character of the jobs, different processing times, different values q_i , or we do not consider the precedence relations, we have a relative error of $O(\log n)$ (see Theorems 2–4).

Now the question arises why do we have ‘good’ lower and upper bounds for the subcases of the problem, but not for the general case? We can try to prove that the RCPSP (with a single type of resources) is not in *APX*, but this seems to be not a trivial task. It is known that in the general case, the RCPSP is not in *APX* (Uetz 2002). It can be proved by a reduction from the graph-coloring problem to a special case of the RCPSP with a large number of types of resources. However, this special case is ‘very specific’ and does not correspond to practical situations. So the proof that ‘the special case with a single resource is not in *APX*’ is an important theoretical task.

We conjecture that the problem is so hard since it consists of four different NP-hard problems (Clique, Partition, Bin-packing, Vertex covering) while each of the subcases PMS, UPT, LSPP contains only two of these problems (see Table 2).

6 A badly approximable special case of the RCPSP is NP-hard

In the previous sections of this paper, we have shown that for the following special case of the RCPSP known upper and lower bounds have a bad approximation ratio (relative error). For this case, we have only two sets of jobs: a set of *high* jobs N_{high} , for which $p_j = \varepsilon$, $q_j = n = Q$, and a set of *long* jobs N_{long} with arbitrary processing times and $q_j = 1$. In addition, we assume that there are no direct precedence relations between the jobs from N_{long} but only precedence relations of the type $i \rightarrow k \rightarrow j$, where $i, j \in N_{long}$ and $k \in N_{high}$.

We denote this case as *batch-case of the RCPSP*. Next, we prove that this special case is NP-hard in the strong sense by a reduction from the single machine parallel-batch scheduling problem $1|p - \text{batch}, \text{chains}, b = \infty|C_{\max}$ which is as follows.

Problem $1|p - \text{batch}, \text{prec}, b = \infty|C_{\max}$:

Let a set $N = \{1, 2, \dots, n\}$ of jobs be given. Job $j \in N$ has to be processed for $p_j \geq 0$ time units without preemption. Furthermore, finish-start precedence relations $i \rightarrow j$ are defined between the jobs according to an acyclic directed graph $G = (N, V)$.

The jobs are processed in batches, where a batch is a subset of jobs, and we require that the batches form a partition of the set of all jobs. The processing time of a batch is equal to the maximum processing time among the jobs in this batch. The completion time of all jobs in a batch is equal to the completion time of the batch. If there is a precedence relation between jobs i and j , then these jobs cannot be processed in the same batch. Moreover, if $i \rightarrow j$, then the starting time of the batch which includes job j must be greater than or equal to the completion time of the batch which includes job i . All jobs in the same batch start simultaneously at the earliest possible starting time.

In Cheng et al. (2004), it has been shown that even for the problem $1|p - \text{batch}, \text{chains}$, $b = \infty|C_{\max}$ with chains as precedence relations, this problem is NP-hard in the strong sense by a reduction from the graph problem *Vertex cover*.

Theorem 5 *The batch-case of the RCPSP is NP-hard in the strong sense.*

Proof Assume that we have an instance of the problem $1|p - \text{batch}, \text{prec}, b = \infty|C_{\max}$. We construct an instance of the batch-case as follows. All jobs $1, 2, \dots, n$ from the initial instance correspond to the jobs $1, 2, \dots, n \in N_{\text{long}}$ in the instance of the batch-case of the RCPSP. The processing times are the same and we have $q_j = 1$ for $j = 1, 2, \dots, n$. In addition, we construct a subset N_{high} . If there is a precedence relation $i \rightarrow j$, $i, j \in N$, then we add a high job k_{ij} with processing time $p_{k_{ij}} = 1$ and $q_{k_{ij}} = n$ and the precedence relations $i \rightarrow k_{ij} \rightarrow j$, $i, j \in N_{\text{long}}$. Moreover, let $Q = n$.

If C_{\max}^{p*} is the optimal value for the initial instance, then $C_{\max}^* = C_{\max}^{p*} + |V|$ is the optimal value for the instance of the batch-case, where $|V|$ is the number of precedence relations in the initial instance.

Let π be an optimal schedule for the instance of the batch-case and $N_{ij} \subset N_{\text{long}}$, $i, j \in N_{\text{long}}$ be the set of long jobs which are processed between two high jobs i and j (there is no other job $k \in N_{\text{high}}$, which is processed between them). Then, in an optimal schedule for the initial instance, all jobs corresponding to jobs from N_{long} will be processed in the same batch. \square

7 Concluding remarks

In this paper, we give some arguments that already a special case of this problem with a single type of resources is not approximable in polynomial time with an approximation ratio bounded by a constant. In particular, we have proven that there exist instances for which the best known lower bound of Mingozzi et al. has a bad relative error of at least $O(\log n)$ and that the calculation of this bound is an NP-hard problem. Moreover, there exists a type of instances for which known ‘polynomial’ upper bounds have approximation ratios of at least $O(\sqrt{n})$ and known lower bounds have a relative error of at least $O(\log n)$. This type of instances corresponds to the problem $1|p - \text{batch}, b = \infty|C_{\max}$, i.e., we have found one of the most difficult subcases of the RCPSP.

Acknowledgements Partially supported by DAAD (Deutscher Akademischer Austauschdienst): A/08/80442/Ref. 325.

References

Bianco, L., & Caramia, M. (2011). A new lower bound for the resource-constrained project scheduling problem with generalized precedence relations. *Computers & Operations Research*, 38, 14–20.

- Brucker, P., & Knust, S. (2006). *Complex scheduling*. Berlin, Heidelberg: Springer.
- Brucker, P., Knust, S., Schoo, A., & Thiele, O. (1998). A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *107*, 272–288.
- Carlier, J., & Neron, E. (2003). On linear lower bounds for the resource constrained project scheduling problem. *European Journal of Operational Research*, *149*, 314–324.
- Cheng, T. C. E., Ng, C. T., Yuan, J. J., & Liu, Z. H. (2004). Single machine parallel batch scheduling subject to precedence constraints. *Naval Research Logistics*, *57*(7), 314–324.
- Demeulemeester, E. L., & Herroelen, W. S. (1996). An efficient optimal solution procedure for the preemptive resource-constrained project scheduling problem. *European Journal of Operational Research*, *90*, 334–348.
- Gafarov, E. R., Lazarev, A. A., & Werner, F. (2010). On lower and upper bounds for the resource-constrained project scheduling problem. Preprint 08/10, FMA, Otto-von-Guericke-Universität Magdeburg.
- 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*, 394–407.
- Kolisch, R., & Hartmann, S. (1998). Heuristic algorithms for solving the resource-constrained project scheduling problem—classification and computational analysis. In J. Weglarz (Ed.), *Project scheduling: recent models, algorithms and applications* (pp. 147–178). Boston: Kluwer.
- Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: an update. *European Journal of Operational Research*, *174*(1), 23–37.
- Kolisch, R., & Padman, R. (2001). An integrated survey of deterministic project scheduling. *Omega*, *29*(3), 249–272.
- Kolisch, R., & Sprecher, A. (1996). PSPLIB—a project scheduling problem library. Manuskripte aus den Instituten für Betriebswirtschaftslehre, No. 396. Kiel, Germany.
- Lawler, E. L., Lenstra, J. K., Rinnooy, Kan A. H. G., & Shmoys, D. B. (1989). *Sequencing and scheduling: algorithms and complexity* (Report BS-R8909). Centre for Mathematics and Computer Science, Amsterdam.
- Lazarev, A. A., & Gafarov, E. R. (2008). On project scheduling problem. *Automation and Remote Control*, *69*(12), 2070–2087.
- Martello, S., Monaci, M., & Vigo, D. (2003). An exact approach to the strip-packing problem. *INFORMS Journal on Computing*, *15*(3), 310–319.
- Mingozzi, A., Maniezzo, V., Ricciardelli, S., & Bianco, L. (1998). An exact algorithm for project scheduling with resource constraints based on new mathematical formulation. *Management Science*, *44*, 714–729.
- Rykov, I. (2006). *Approximate solving of RCPSP*. Abstract Guide of OR 2006, Karlsruhe 6.9.-8.9.2006, Germany, p. 226.
- Steinberg, A. (1997). A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, *26*(2), 401–409.
- Uetz, M. (2002). *Algorithms for deterministic and stochastic scheduling*. Ph.D. thesis. Cuvillier Verlag.