On Lower and Upper Bounds for the Resource-Constrained Project Scheduling Problem

Evgeny R. Gafarov [a], **Alexander A. Lazarev** [b]

Institute of Control Sciences of the Russian Academy of Sciences,
Profsoyuznaya st. 65, 117997 Moscow, Russia,
email: [a] *axel73@mail.ru,* [b] *jobmath@mail.ru*

Frank Werner

Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg,
PSF 4120, 39016 Magdeburg, Germany,
email: frank.werner@mathematik.uni-magdeburg.de

March 3, 2010

Abstract

In this paper, we consider the well-known resource-constrained project scheduling problem. We propose 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 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 $1|p-batch, b=\infty|C_{max}$.

Keywords: Project scheduling, Makespan, Lower bounds, Upper bounds

MSC classification: 90 B 35

1 Introduction

Problem RCPSP may be formulated as follows. Given a set $N = \{1, 2, \ldots, n\}$ of jobs. A constant amount (quantity) of $Q_k > 0$ units of resource $k, k = 1, 2, \ldots, K$, is available at 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, $j = 1, 2, \ldots, n$, 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 [1], 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 (e.g. [2]). There exists a special electronic library PSPLIB [3] 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. [1]. An experimental analysis of some heuristics has been done in [5] and [6, 7]. A solution algorithm for the special case with preemptions has been proposed in [8]. 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 [14]. However, this special case is rather specific and does not correspond to practical situations.

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 $p_0 = p_{n+1} = 0$, and there are precedence relations between

each triplet of jobs $0 \to j \to n+1,\ j = 1, 2, \ldots, n,\ q_{0k} = q_{n+1,k} = 0$ for each resource $k = 1, 2, \ldots, K$. Let UB be an upper bound for 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:

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

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

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 $\tau,\ k = 1, 2, \ldots, K$.
2. If $EL = \emptyset$, then goto 10;
3. Choose one job $j \in EL$;
4. If $j = 0$, then $t := 0$ else $t := \max_{i|i\to j}\{S_i + p_i\}$;
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 $Q_k(\tau) := Q_k(\tau) - q_{jk},\ k = 1, 2, \ldots, K,$
   $\forall \tau \in [t, t + p_j)$;
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, then we consider the *intervals* $[t, t]$ and $[S_j, C_j] = [t, t]$ under the assumption that $q_{jk} = 0$ for $k = 1, 2, \ldots, K$.

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.

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

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 Sections 2 and 3, we analyze upper and lower bounds for special cases of the RCPSP. Then, in Sections 4 and 5, upper and lower bounds for the general case are investigated. In Section 6, 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 Section 7.

2 Upper Bounds for Special Cases

In this section, we consider three special cases of the RCPSP. We denote by $C_{max}(LS_a)$ the value of the objective function for a schedule

obtained by Algorithm LS according to dispatching rule a. The dispatching rule is used to select a job in Step 3 of Algorithm LS. We remind that the list of the used dispatching rules is presented in Table 1.

1. **PMS** (Parallel machine scheduling). For this case, we have $q_i = 1$ for $i = 1, 2, \ldots, n$ and $Q_1 = m \in Z$, where m is the number of machines. This problem is NP-hard in the strong sense [9]. For this special case, it is known that

$$\frac{C_{\max}(LS)}{C^*_{\max}} < 2$$

for any dispatching rule [9].

2. **LSPP** (like a Strip packing problem). For this case, there are no precedence relations. It is obvious that this special case of the RCPSP is similar to the strip packing problem which is as follows. **Strip packing problem.** Given a horizontal strip of height Q_1 and a list L of rectangular pieces $\{1, 2, \ldots, n\}$, pack the pieces into the strip such that the width 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 its height q_{i1}.

   Problem SPP is NP-hard in the strong sense. The relation between the special case LSPP and the strip packing problem is obvious. Let SPP^* be the optimal width of the strip for the strip packing problem. In [12], it has been shown that $LSPP^* \leq SPP^*$ (not always $LSPP^* = SPP^*$), where $LSPP^*$ is the optimal value for the corresponding instance of LSPP. It is known [12] that the following relation holds:

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

   For LSPP, we can prove that

$$\frac{C_{\max}(LS_{DEST})}{C^*_{\max}} < 3$$

   (see Theorem 1).

3. **UPT** (Unit processing time). Here, we have $p_i = 1$ for $i = 1, 2, \ldots, n$. This special case is also NP-hard in the strong sense [9]. For this case, we can prove that

$$\frac{C_{\max}(LS_{EST})}{C^*_{\max}} < 4$$

   (see Theorem 2).

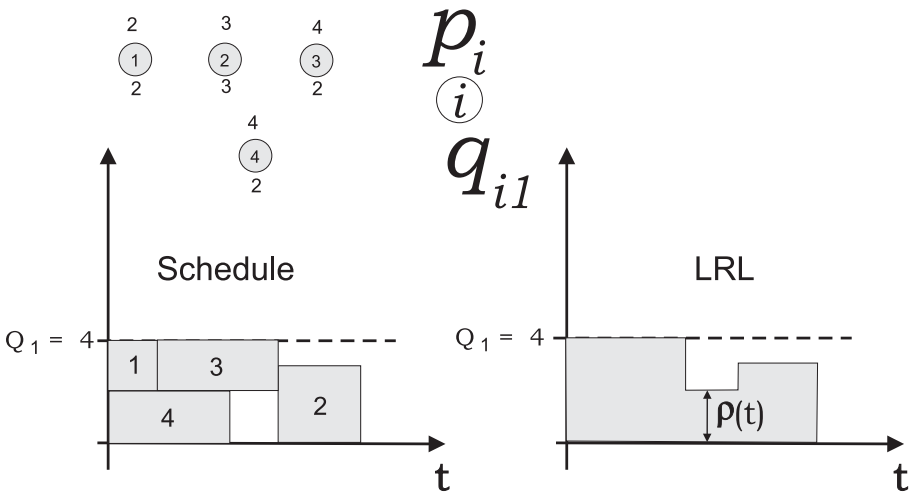

Figure 1: The level of the load $\rho(t)$ of the resource

2.1 Upper Bound for the Special Case LSPP

For the special case LSPP, we consider Algorithm LS using the dispatching rule *DEST*: In Step 3 of Algorithm LS, we choose a job with the earliest starting time t, at which one can begin to process this job without violating the precedence relations or resource constraints.

Theorem 1 *For the special case LSPP, inequality*

$$\frac{C_{\max}(LS_{DEST})}{C^*_{\max}} < 3$$

holds.

Proof. Denote by $\rho(t)$ (see Fig. 1) the level of the load of the resource at the time point t, $\rho(t) = \sum_{i \in N_t} q_i$, where N_t is the set of jobs that are processed at time t.

Denote $UB = C_{\max}(LS_{DEST})$. Let $[t_1, t_2] \in [0, UB)$ be an interval of minimal length for which we have $\rho(t) > \frac{Q_1}{2}$ for all $t \in [0, t_1)$ and $\rho(t) > \frac{Q_1}{2}$ for all $t \in (t_2, UB)$. It is obvious that for the schedule obtained by Algorithm LS with the dispatching rule DEST, we have $t_2 - t_1 \leq p_{max}$. Otherwise, if $t_2 - t_1 > p_{max}$, then there is a job $j \in N_{t_2}$ with $q_j \leq \frac{Q_1}{2}$ and $S_j > t_1$, so we get a contradiction since, according to dispatching rule DEST, we must process this job from time t_1. Thus, we have $t_2 - t_1 \leq p_{max}$. Therefore, $UB - p_{max} < 2 \cdot \frac{\sum_{i=1}^n q_i p_i}{Q_1}$. In addition, we have

$$C^*_{max} \geq \frac{\sum_{i=1}^n q_i p_i}{Q_1}$$

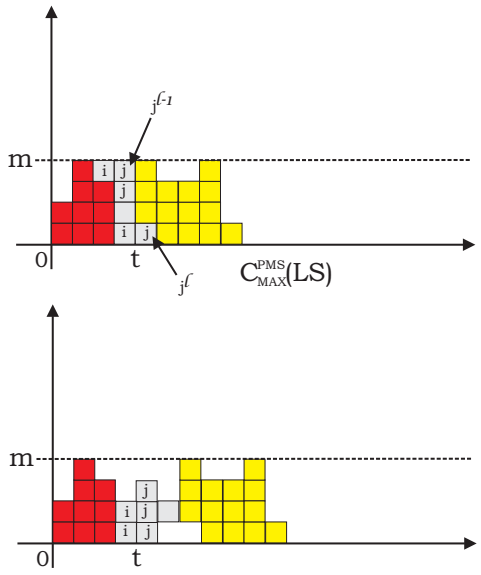

Figure 2: Special case UPT with $C_{max}(LS_{EST})/C^*_{max} < 4$

and

$$C^*_{max} \geq p_{max}.$$

Thus, $UB < 3C^*_{max}$.

□

2.2 Upper Bound for the Special Case UPT

Without loss of generality, assume that $Q_1 \in Z$ and $q_i \in Z$, $i = 1, 2, \ldots, n$. For this special case, we propose the following algorithm for calculating an upper bound. We split each job $j = 1, 2, \ldots, n$, with $q_j > 1$ into several jobs $j^1, j^2, \ldots, j^{q_j}$ with processing times 1 and $q_l = 1$, $l = j^1, j^2, \ldots, j^{q_j}$. If $j \to i$, then we assume $j^l \to i^e, l = 1, 2, \ldots, q_j$, $e = 1, 2, \ldots, q_i$. We denote $m = Q_1$.

Now we have a new instance which corresponds to the special case PMS. For this case, we have $\frac{C_{\max}(LS)}{C^*_{\max}} < 2$ for any dispatching rule. For this instance, we construct a schedule according to the following dispatching rule using Algorithm LS: Choose the job with the earliest starting time r_j. If there are several such jobs, then job j^l has the priority, if we have chosen job j^{l-1} in the previous step.

Then there can be a situation when two jobs j^{l-1} and j^l, $2 \leq l \leq q_j$, are processed at time points t and $t+1$ (see Fig. 2). We can construct a feasible solution for the initial instance according to the modification

illustrated in Fig. 2. Moreover, we have $2 \cdot C_{max}^{PMS}(LS) > C_{max}^{UPT}$, where $C_{max}^{PMS}(LS)$ is the value of the objective function for the instance for PMS obtained by Algorithm LS and C_{max}^{UPT} is the value of the objective function for a feasible solution of the initial instance obtained according to the modification from Fig. 2. The latter value can be obtained by Algorithm LS using the dispatching rule EST. Denote by $C_{\max}^{PMS*}$ and $C_{\max}^{UPT*}$ the optimal values of the objective function for these instances. Then we have $C_{\max}^{PMS*} \leq C_{\max}^{UPT*}$ and

$$2 \cdot (2 \cdot C_{\max}^{PMS*}) > 2C_{max}^{PMS}(LS) > C_{max}^{UPT} \geq C_{max}^{UPT*}.$$

Thus, the following theorem holds.

Theorem 2 *For the special case UPT, inequality*

$$\frac{C_{\max}(LS_{EST})}{C_{\max}^{*}} < 4$$

holds.

Moreover, we have

$$C_{max}^{PMS}(LS) < \frac{\sum_{i=1}^{n} q_i}{Q_1} + CP \qquad [9],$$

thus, we obtain the following remark.

Remark 1 *For UPT, we have*

$$C_{max}^{UPT} < 2 \cdot C_{max}^{PMS}(LS) < 2\left(\frac{\sum_{i=1}^{n} q_i}{Q_1} + CP\right).$$

Thus, all three special cases belong to APX.

3 Lower Bounds for Special Cases

For the three special cases mentioned above, the following results are known:

1. PMS (Parallel machine scheduling). For this case, we have

$$C_{\max}^{*} < \frac{\sum_{i=1}^{n} p_i}{m} + CP \qquad [9],$$

where $\frac{\sum_{i=1}^{n} p_i}{m}$ and CP (i.e., the length of a critical path) are lower bounds for this problem.

2. LSPP (like a Strip packing problem). For the strip packing problem, it is known that

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

The proof was given in [10] (which is based on a result from [11]).

3. UPT (Unit processing time). According to Remark 1, we have

$$C^*_{\max} < 2 \left(\frac{\sum_{i=1}^{n} q_i}{Q_1} + CP \right),$$

where $\frac{\sum_{i=1}^{n} q_i}{Q_1}$ and CP (i.e., the length of a critical path) are lower bounds for this problem.

So for each special case, the simple lower bound

$$LB = \max \left\{ \frac{\sum_{i=1}^{n} q_i p_i}{Q_1}, CP \right\}$$

has a relative error bounded by a constant.

4 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 [1]. Here we show that for each of these dispatching rules, there exists an instance for which we have

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

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

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

Proof. We consider the instance from Fig. 3 (SPT, LPT). We have h chains of jobs $a_i \rightarrow b_i \rightarrow c_i, \ i = 1, 2, \ldots, h$. For each job $b_i, \ i = 1, 2, \ldots, h$, we have $p_{b_i} = p - \varepsilon(i-1), \ q_{b_i} = 1$. For each job $a_i, \ i = 1, 2, \ldots, h$, we have $p_{a_i} = \varepsilon(h - i + 1), \ q_{a_i} = h$. For each job $c_i, \ i = 1, 2, \ldots, h$, 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 $\pi = (a_1, a_2, \ldots, a_h, b_1, b_2, \ldots, b_h, c_1, c_2, \ldots, c_h)$ of job choices but for the dispatching rule SPT, we get the sequence $\pi_{SPT} =$

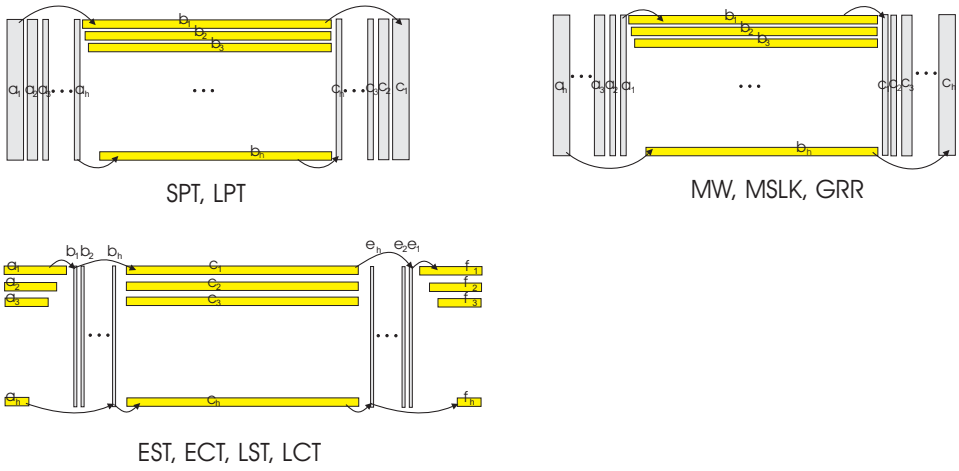

Figure 3: Dispatching rules. Part 1

$(a_h, a_{h-1}, \ldots, a_1, b_h, c_h, b_{h-1}, c_{h-1}, \ldots, b_1, c_1)$ of choices, i.e., all jobs $b_1, b_2, \ldots, b_h$ are processed successively without overlapping. Thus, we get $C^*_{\max} = 2(h + (h-1) + \ldots + 1)\varepsilon + p$ and $C_{\max}(LS_{SPT}) = 2(h + (h-1) + \ldots + 1)\varepsilon + p + (p - \varepsilon) + (p - 2\varepsilon) + \ldots + (p - (h-1)\varepsilon)$. Therefore, we obtain

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

In an analogous way, we may consider the sequence $\pi_{LPT} = (a_1, b_1, c_1, a_2, b_2, c_2, \ldots, a_h, b_h, c_h)$ 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.

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

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

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. 3 (EST, ECT, LST, LCT). We have h chains of jobs $a_i \to b_i \to c_i \to e_i \to f_i,\ i = 1, 2, \ldots, h$. For each job $b_i, e_i,\ i = 1, 2, \ldots, h$, we have $p_{b_i} = p_{e_i} = \varepsilon, q_{b_i} = q_{e_i} = h$. For each job $a_i, f_i,\ i = 1, 2, \ldots, h$, we have $p_{a_i} = p_{f_i} = 2\varepsilon(h - i + 1),\ q_{a_i} = q_{f_i} = 1$. For each job $c_i,\ i = 1, 2, \ldots, h$, we have $p_{c_i} = p,\ q_{c_i} = 1$, where $h^2\varepsilon \ll p$ and $Q_1 = h$.

For this instance, an optimal schedule corresponds to the sequence $\pi = (a_1, a_2, \ldots, a_h, b_1, b_2, \ldots, b_h, c_1, c_2, \ldots, c_h, e_1, e_2, \ldots, e_h, f_1, f_2, \ldots, f_h)$ of job choices but for the dispatching rule EST, we get the sequence $\pi_{EST} = (a_1, a_2, \ldots, a_h, b_h, c_h, e_h, f_h, b_{h-1}, c_{h-1}, e_{h-1}, f_{h-1}, \ldots, b_1, c_1, e_1, f_1)$ of job choices, i.e., all jobs $c_1, c_2, \ldots, c_h$ are processed successively without overlapping. Therefore, we obtain

$$\lim_{\varepsilon \to 0} \frac{C_{\max}(LS_{EST})}{C^*_{\max}} = h = \frac{n}{5}.$$

In an analogous way, we may consider the sequence $\pi_{LST} = (a_1, b_1, c_1, e_1, f_1, \ldots, a_h, b_h, c_h, e_h, f_h)$ of job choices.

□

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

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

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 from Fig. 3 (EST, ECT, LST, LCT). We have h chains of jobs $a_i \to b_i \to c_i \to e_i \to f_i,\ i = 1, 2, \ldots, h$. For each job $b_i, e_i,\ i = 1, 2, \ldots, h$, we have $p_{b_i} = p_{e_i} = \varepsilon, q_{b_i} = q_{e_i} = h$. For each job $a_i, f_i,\ i = 1, 2, \ldots, h$, we have $p_{a_i} = p_{f_i} = 2\varepsilon(h - i + 1),\ q_{a_i} = q_{f_i} = 1$. For each job $c_i,\ i = 1, 2, \ldots, h$, we have $p_{c_i} = p - 4(i - 1)\varepsilon,\ q_{c_i} = 1$, where $h^2\varepsilon \ll p$ and $Q_1 = h$.

For this instance, an optimal schedule corresponds to the sequence $\pi = (a_1, a_2, \ldots, a_h, b_1, b_2, \ldots, b_h, c_1, c_2, \ldots, c_h, c_k, e_1, e_2, \ldots, e_h, f_1, f_2, \ldots, f_h)$ of job choices while for the dispatching rule ECT, we get the sequence $\pi_{ECT} = (a_h, b_h, a_{h-1}, b_{h-1}, \ldots, a_1, b_1, c_h, e_h, f_h, c_{h-1}, e_{h-1}, f_{h-1}, \ldots, c_1, e_1, f_1)$ of selecting the jobs, i.e., all jobs $c_1, c_2, \ldots, c_h$ are processed successively without overlapping. Therefore, we obtain

$$\lim_{\varepsilon \to 0} \frac{C_{\max}(LS_{ECT})}{C^*_{\max}} = h = \frac{n}{5}.$$

In analogous way, we may consider the sequence $\pi_{LCT} = (a_1, b_1, c_1, e_1, f_1, \ldots, a_h, b_h, c_h, e_h, f_h)$ of selecting the jobs.

□

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

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

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

For this instance, an optimal schedule corresponds to the sequence $\pi = (a_1, a_2, \ldots, a_h, b_1, b_2, \ldots, b_h, c_1, c_2, \ldots, c_h)$ of selecting the jobs while for the dispatching rule MW, we get the sequence $\pi_{SPT} = (a_h, a_{h-1}, \ldots, a_1, b_h, c_h, b_{h-1}, c_{h-1}, \ldots, b_1, c_1)$ of job choices since $d_{a_h} - r_{a_h} = \ldots = d_{a_1} - r_{a_1} = UB - (p + h\varepsilon)$, $d_{b_h} - r_{b_h} = UB - 2h\varepsilon < d_{b_{h-1}} - r_{b_{h-1}} = UB - 2(h-1)\varepsilon < \ldots < d_{b_1} - r_{b_1}$ and $d_{c_h} - r_{c_h} = UB - p - h\varepsilon < d_{b_{h-1}} - r_{b_{h-1}} < \ldots < d_{b_1} - r_{b_1}$, and so on. Thus, all jobs $b_1, b_2, \ldots, b_h$ are processed successively without overlapping. Therefore, we obtain

$$\lim_{\varepsilon \to 0} \frac{C_{\max}(LS_{MW})}{C^*_{\max}} = h = \frac{n}{3}.$$

In an analogous way, we may consider the sequence $\pi_{MSLK} = (a_h, b_h, c_h, a_{h-1}, b_{h-1}, c_{h-1}, \ldots, a_1, b_1, c_1)$ for selecting the jobs, 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 $i = 1, 2, \ldots, h$. Similarly, we may consider the sequence $\pi_{GRR} = (a_1, a_2, \ldots, a_h, b_h, c_h, b_{h-1}, c_{h-1}, \ldots, b_1, c_1)$.

□

Lemma 5 *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. 4 (MIS, LIS). We have h independent sets of jobs, each consists of a chain $b_i \to c_i$, and subsets $DB_i, DC_i,\ i = 1, 2, \ldots, h$, where $b_i \to DB_i$ and $c_i \to DC_i$. The notation $j \to X_i$ means that for all $l \in X_i$, we have $j \to l$. In addition, we have $|DB_i| = (i - 1)$, $|DC_i| = i$ for $i = 1, 2, \ldots, h$. For all jobs l from these subsets, we have $p_l = \varepsilon$, $q_l = \varepsilon$.

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

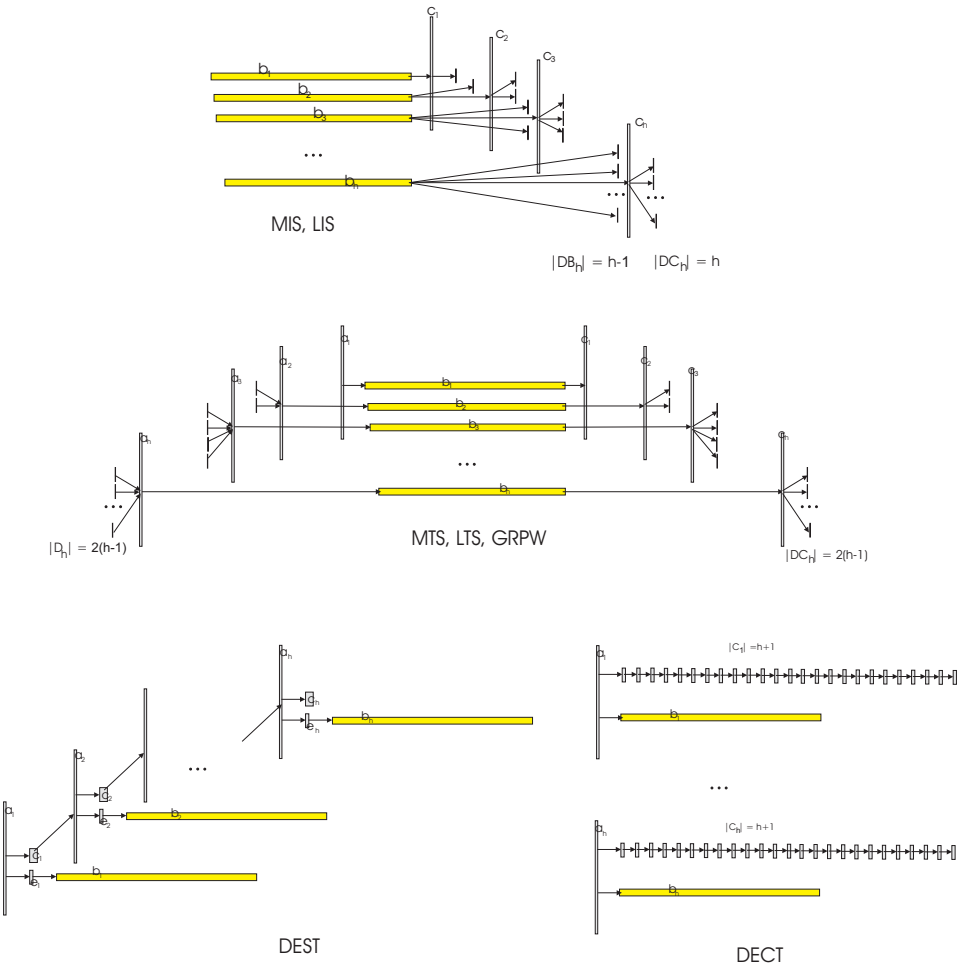

Figure 4: Dispatching rules. Part 2

As before, we see that in an optimal schedule, all jobs b_i are processed in parallel while in schedule π_{MIS}, these jobs are processed successively without overlapping. Therefore, we obtain

$$\lim_{\varepsilon \to 0} \frac{C_{\max}(LS_{MIS})}{C^*_{\max}} = h.$$

We have $n = h + (2 + 4 + 6 + \ldots + 2h) = h + 2(1 + 2 + 3 + \ldots + h) = h + h(h + 1)$. Therefore, $h = O(\sqrt{n})$.

In an analogous way, we may consider an instance, where $p_{b_i} = p - (h - i)\varepsilon,\ i = 1, 2, \ldots, h$, and the sequence π_{LIS} of selecting the jobs.

As mentioned before, it is easy to construct an analogous instance with a symmetric graph of precedence relations (see the idea in the proof of Theorem 3 later).

□

Lemma 6 *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 5. For an illustration, see Fig. 4 (MTS, LTS, GRPW), where $|DC_i| = |D_i| = 2(i - 1),\ i = 1, 2, \ldots, h$.

□

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

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

Proof. We consider 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.

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

For this instance, an optimal schedule corresponds to the sequence $\pi = (a_1, c_1, e_1, a_2, c_2, e_2, \ldots, a_h, c_h, e_h, b_1, b_2, \ldots, b_h)$ of job choices while for dispatching rule DEST, we get the sequence $\pi_{DEST} =$

$(a_1, c_1, e_1, b_1, \ldots, a_h, c_h, e_h, b_h)$, i.e., all jobs $b_1, b_2, \ldots, b_h$ are processed successively without overlapping. Therefore, we obtain

$$\lim_{\varepsilon \to 0} \frac{C_{\max}(LS_{DEST})}{C^*_{\max}} = h = \frac{n}{4}.$$

As mentioned before, it is easy to construct an analogous instance with a symmetric graph of precedence relations.

□

Lemma 8 *There exists an instance of the RCPSP for which*

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

Proof. We consider 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.

We consider the instance from Fig. 4 (DECT). We have h independent sets of jobs such that the i-th set contains a set of jobs $\{a_i, b_i\} \bigcup C_i$, where $|C_i| = h + 1,\ C_i = \{j^i_1, \ldots, j^i_{h+1}\}, i = 1, 2, \ldots, h$. For each job $b_i,\ i = 1, 2, \ldots, h$, we have $p_{b_i} = p,\ q_{b_i} = 1$. For each job $a_i,\ i = 1, 2, \ldots, h$, we have $p_{a_i} = 2\frac{p}{h},\ q_{a_i} = h + 1$. For each job $c \in C_i,\ i = 1, 2, \ldots, h$, we have $p_c = \frac{p}{h},\ q_c = \frac{1}{h}$. Moreover, let $Q_1 = h + 1$.

For this instance, an optimal schedule corresponds to the sequence $\pi = (a_1, a_2, \ldots, a_h, b_1, b_2, \ldots, b_h, j^1_1, j^2_1, \ldots, j^h_1, \ldots, j^1_{h+1}, j^2_{h+1}, \ldots, j^h_{h+1})$ of selecting the jobs while for dispatching rule DECT, we get the sequence $\pi_{DECT} = (a_1, j^1_1, j^1_2, \ldots, j^1_{k+1}, b_1, a_2, \ldots, b_2, a_3, \ldots)$ of job choices, i.e., all jobs $b_1, b_2, \ldots, b_h$ are processed successively without overlapping.

Therefore, we obtain

$$\lim_{n \to \infty} \frac{C_{\max}(LS_{DECT})}{C^*_{\max}} = h.$$

We have $n = 2h + (h + 1)h$, i.e., $h = O(\sqrt{n})$ holds.

□

Moreover, we can present the following result.

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

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

for all dispatching rules a mentioned above.

Proof.

It is easy to construct such an instance. This instance contains the subsets of jobs $N_1, N_2, \ldots, N_{10}$. Each subset of jobs $N_i, i = 1, 2, \ldots, 10$ corresponds to the instance from Lemmas 1 - 8 (in Lemmas 5 and 6, we have altogether four instances). In each subset $N_i, i = 1, 2, \ldots, 10$, we have h jobs with *long* processing times approximately equal to p. In addition, we have 9 dummy jobs $o_1, o_2, \ldots, o_9$ such that for all $j \in N_i$ and $l \in N_{i+1},\ i = 1, 2, \ldots, 9$, we have $j \to o_i \to l$.

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.

□

5 Lower Bounds for the General Case

From [13], it is known that existing *polynomial* lower bounds for this problem have *bad* relative errors.

Denote $LB_0 = CP$, where CP is the length of a critical path in graph G describing the precedence relations. It is evident that LB_0 is a lower bound for the optimal objective function value $C^*_{\max}$.

Lemma 9 *[13] There exists an instance of the RCPSP for which*

$$\frac{C^*_{\max}}{LB_0} = n.$$

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

Lemma 10 *[13] There exists an instance of the RCPSP for which*

$$\frac{C^*_{\max}}{LB_1} = n.$$

Let

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

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 jobs from CP without violating the resource constraints.

Lemma 11 *[13] There exists an instance of the RCPSP for which*

$$\frac{C^*_{\max}}{LB_S} = \frac{n}{2}.$$

In [4, 14, 15, 16], specific algorithms have been presented for the calculation of lower bounds. One of these bounds is the lower bound given by Mingozzi.

5.1 Lower Bound by Mingozzi et al.

We show that the calculation of the lower bound by Mingozzi et al. [4], LB_M, is an NP-hard problem. 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 $k = 1, 2, \ldots, K$). 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 $i = 1, 2, \ldots, n$.

We denote all these sets by $X_1, X_2, \ldots, X_f$, 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, $j = 1, 2, \ldots, f$.

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:

$$\sum_{j=1}^{f} x_j \longrightarrow \min \tag{1}$$

$$\begin{cases} \sum\limits_{j=1}^{f} a_i^j x_j \geq p_i, \ i = 1, 2, \ldots, n; \\ x_j \geq 0, \ j = 1, 2, \ldots, f. \end{cases} \tag{2}$$

Lemma 12 *The calculation of LB_M is an NP-hard problem.*

Proof. Consider the NP-hard problem "Packing in Pallets". We have n items with lengths w_i, $i = 1, 2, \ldots, n$, and some pallets with length W. The objective is to pack the items in the pallets in such a way that the number of pallets used is minimized.

We consider a special case of the $RCPSP$ and its reduction from the "Packing in Pallets" problem. Each job i, $i = 1, 2, \ldots, n$, corresponds to an item i. Let $p_i = 1$, $q_{i1} = w_i$ for each job $i = 1, 2, \ldots, n$ and $Q_1 = W$. There are no precedence relations. Then $C^*_{\max}$ is equal to the minimum number of pallets.

Obviously, $LB_M = C^*_{\max}$ for this special case of the $RCPSP$. Thus, the calculation of LB_M is an NP-hard problem.

□

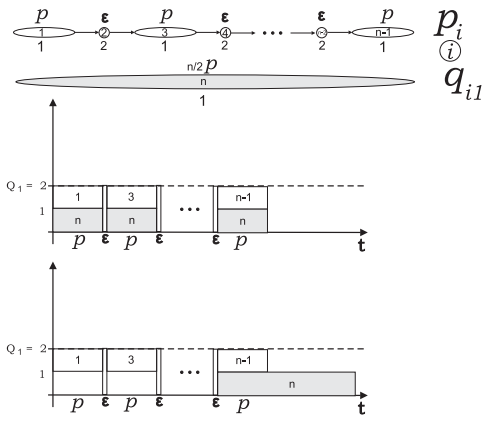

Figure 5: An instance for illustrating Lemma 13

Lemma 13 *There exists an instance of the RCPSP for which*

$$\frac{C^*_{\max}}{LB_M} \approx 2.$$

Proof. Consider the instance of the *RCPSP* shown in Fig. 5. For this instance, we have the following sets X_j:

$$X_1 = \{1, n\}, \quad X_2 = \{2\}, \quad X_3 = \{3, n\},$$
$$X_4 = \{4\}, \quad \ldots, \quad X_{n-1} = \{n-1, n\},$$

and the optimal solution of the corresponding LP is

$$x_1 = x_3 = \ldots = x_{n-1} = p, \quad x_2 = x_4 = \ldots = x_{n-2} = \varepsilon \qquad [4].$$

Then we have

$$\frac{C^*_{\max}}{LB_M} = \frac{\frac{n}{2}p + \frac{n-2}{2}\varepsilon + (\frac{n}{2}-1)p}{\frac{n}{2}p + \frac{n-2}{2}\varepsilon} = \frac{p + \varepsilon - \frac{2}{n}\varepsilon + p - \frac{2}{n}p}{p + \varepsilon - \frac{2}{n}\varepsilon}$$

Thus, we obtain

$$\lim_{n\to\infty,\ \varepsilon\to 0} \frac{C^*_{\max}}{LB_M} = 2,$$

i.e., we can vary the parameters $n,\ p,\ \varepsilon$ such that

$$\frac{C^*_{\max}}{LB_M} \approx 2.$$

□

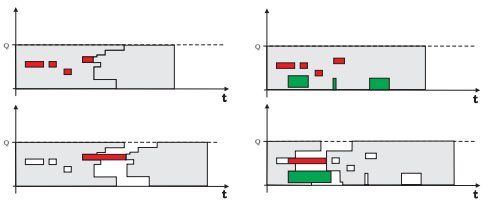

Figure 6: One or two preempted jobs

5.2 The Conjecture $C^*_{\max} < 2 \cdot C^*_{\max}(pmtn)$

In [13], there is a conjecture that $C^*_{\max} < 2 \cdot C^*_{\max}(pmtn)$. This conjecture is true for some special cases. However, the conjecture is false for the general case. First, we propose several special cases, for which the conjecture holds.

1. It is known that the conjecture is true for the special case of problem $Pm|prec|C_{\max}$, since

$$C^*_{\max} \leq \frac{\sum_{i=1}^{n} q_{i1} \cdot p_i}{Q_1} + CP \leq 2C^*_{\max}(pmtn) \quad [9].$$

2. For the special case LSPP, the conjecture is true. The proof is based on a result from [10] (which is based on a result from [11]) for the strip packing problem.

   From [10], it is known that

$$2 \max \left\{ \frac{\sum_{i=1}^{n} q_{i1} \cdot p_i}{Q_1}, p_{max} \right\} > C^*_{max}.$$

   Since

$$C^*_{\max}(pmtn) \geq \frac{\sum_{i=1}^{n} q_{i1} \cdot p_i}{Q_1} \quad \text{and} \quad C^*_{\max}(pmtn) \geq p_{max},$$

   the conjecture is true.

3. If there are only one or two preempted job in an optimal schedule for the preemptive RCPSP, then the conjecture is true (see Fig. 6). For example, if there is only one preempted job in an optimal schedule for the problem with preemptions (left-upper figure), then we can transform this schedule into a feasible schedule for the problem without preemptions (left-down figure).

4. If all preempted jobs are processed successively without overlapping (i.e., there is no interval $[t_1, t_2)$, where two preempted job are processed in parallel), then the conjecture is true.

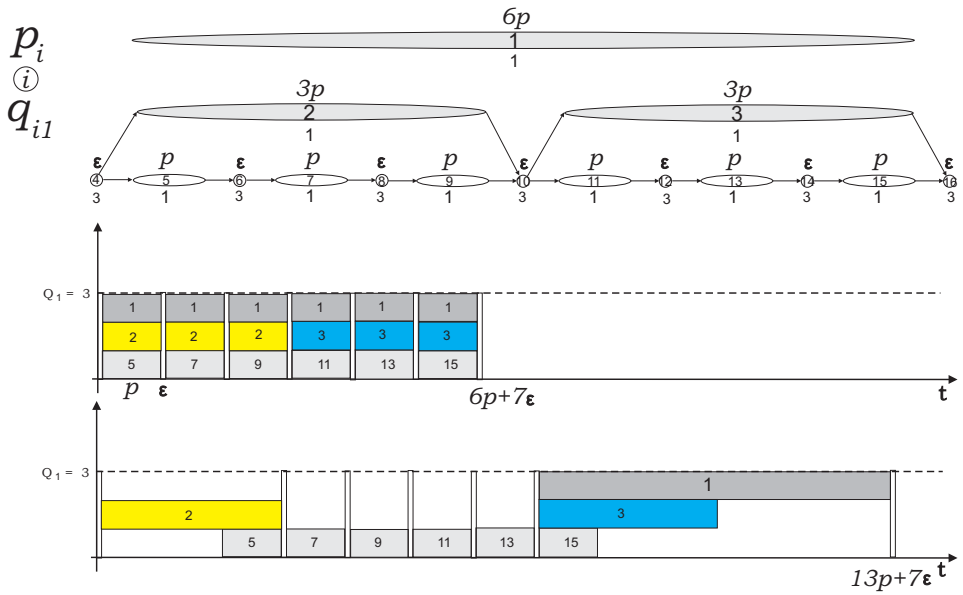

Figure 7: An instance, where $C^*_{\max} > 2 \cdot C^*_{\max}(pmtn)$

However, the conjecture is not true for the special instance given in Fig. 7, where $p \gg \varepsilon$ (e.g. $\varepsilon = 1/(np)^n,\ p > 1$). We can construct an analogous instance, where $2h$ jobs have processing time p, $2h+1$ jobs have a *small* processing time ε, one job has the processing time $2h \cdot p$ and two jobs have the processing time $h \cdot p$, i.e., $n = 4h + 4$. If h is large and ε is very small, we have $C^*_{\max} \approx 2.5 \cdot C^*_{\max}(pmtn)$.

From this instance, we get the following lemma.

Lemma 14 *There exists an instance of the RCPSP for which*

$$\frac{C^*_{\max}}{LB_M} \approx 2.5.$$

As we show in the following, we can construct a similar instance for which even

$$\frac{C^*_{\max}}{LB_M} = O(\log n).$$

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

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

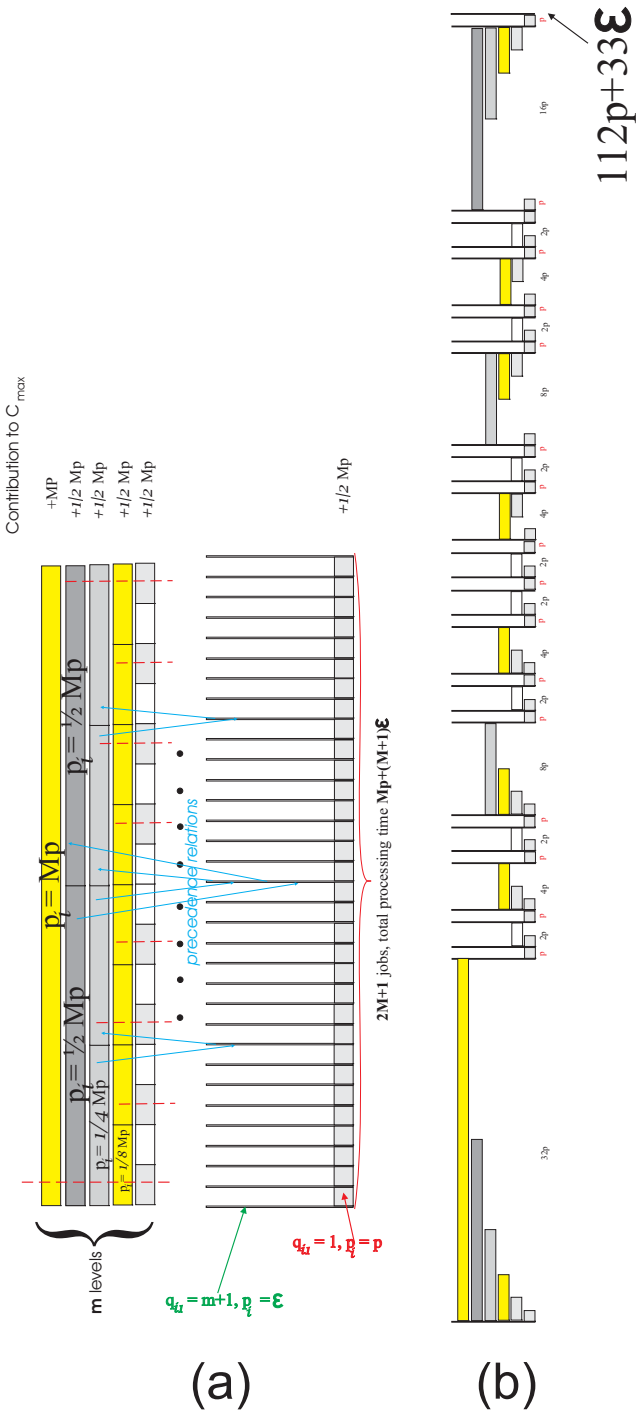

Figure 8: An instance for illustrating Theorem 4

Proof.

We consider an instance of the type given in Fig. 8 (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 $h,\ h = 2,3,\ldots,m$, we have 2^{h-1} jobs 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 $e_1 \to j_1^{m+1} \to e_2 \to j_2^{m+1} \to \ldots \to e_M \to j_M^{m+1} \to e_{M+1}$, where $p_{e_i} = \varepsilon,\ q_{e_i} = m+1,\ i = 1,2,\ldots,M+1$, and $p_{j_i^{m+1}} = p,\ q_{j_i^{m+1}} = 1,\ i = 1,2,\ldots,M$. For each job j_i^h of level $h,\ h = 2,3,\ldots,m$, we have $p_{j_i^h} = \frac{M}{2^{h-1}}p,\ q_{j_i^h} = 1$. At each level $h,\ h = 2,3,\ldots,m$, we have a chain of jobs $e_1 \to j_1^h \to e_{\frac{M}{2^{h-1}}+1} \to j_2^h \to e_{2\frac{M}{2^{h-1}}+1} \to \ldots \to j_{2^{h-1}}^h \to e_{M+1}$. Some of these precedence relations are illustrated in Fig. 8 (a). For this instance, we have $(M+1)\varepsilon \ll p$ (e.g. $\varepsilon = 1/((M+1)p)^2,\ p \in Z_+$). By dotted lines, we separate the jobs which will be processed in parallel in an optimal schedule for the non-preemptive problem.

In Fig. 8 (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

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

Let us now express M by means of m. We have $2^m = M$. Then

$$n = 2M+1+1+2+\ldots+2^{m-1} = 2M+1+2^m-1 = 2\cdot 2^m+2^m = 3\cdot 2^m$$

from which we obtain

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

Therefore, we get

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

□

As a consequence, there exists an instance of the RCPSP for which $\frac{C^*_{\max}}{LB_M} = O(\log n)$, and we can formulate the following result:

Theorem 5 *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. 8 (a), where $Q = m+1$. In set N_2, we have n independent jobs with unit processing times $p_j = 1$ and $\sum_{j \in N_2} q_j = 2m + 2$. Additionally, we have a dummy job o_1 such that $j \to o_1 \to l$ for all $j \in N_1,\ l \in N_2$. It is obvious that we can present a reduction from the partition problem to the problem of calculating LB_M for this type of instances.

5.3 Other Lower Bounds

We can consider the optimal values of the objective function for the special sub-cases PMS, $LSPP$, 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 Section 3, we have shown that $C^{PMS*}_{max} < LB_0 + LB_1 \leq 2LB_M$, $C^{LSPP*}_{max} < 2\max\{LB_0,\ LB_1\} \leq 2LB_M$ and $C^{UPT*}_{max} < 2(LB_0 + LB_1) \leq 4LB_M$ (e.g. C^{LSPP*}_{max} is the optimal value of objective function for the relaxed instance which corresponds to special case LSPP). Therefore, we can prove the following theorem:

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

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

6 Some Further Remarks

First, we give some remarks about the approximability of the RCPSP.

1. If we do not take into consideration non-preemptive character of 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 4 - 6).
2. In some papers, the authors have proposed methods of relaxing the RCPSP in order to calculate a *good* lower bound. For example, we can consider only such jobs for which $q_i > a$, where a is constant. For the instance presented in Fig. 8 and Theorem 4, it appears to be difficult to do this.
3. We can estimate the change in the optimal function value, if we delete some precedence relations from graph G. It is obvious that there exists an instance for which, if we delete some relations from the critical path (i.e., both vertices are incident with an arc that

belongs to the critical path), we have a new instance for which $C'/C^*_{max} = O(h)$, where h is the number of deleted precedence relations and C' is the optimal makespan value for the modified instance.

4. We can prove the following lemma:

   Lemma 15 *Let UB be an upper bound for the objective function value and r_j and $d_j, j = 1, 2, \ldots, n$, as defined in the introduction. Then we have*

   $$UB - C^*_{max} \leq \min\{d_i - r_i - p_i\}.$$

   This fact can be used in practice for the development of algorithms.

Now the question arises why do we have *good* lower and upper bounds for the sub-cases of the problem considered, but not for the general case? We can try to prove that the RCPSP is not in APX, but this does not seem to be simple. Existing methods for proving that a problem is not in APX use the *gap* method. For example, for the traveling salesman problem, we know that there is no polynomial algorithm to compute a feasible solution with a guaranteed approximation ratio. One can construct an instance with any approximation ratio. This result has been proven when there is a matrix of the input-data (for example, the matrix of distances). For the RCPSP, we cannot use such a method. One existing analogous result for the RCPSP with several types of resources can be seen in [14].

We conjecture that the problem is so hard since it contains four different NP-hard problems (see first row in Table 2), while each of the sub-cases contains only two of them (see rows 2-4 in Table 2).

Table 2: NP-hard reducibility

Problem	**Reduction from**
RCPSP with a single resource	Bin-packing problem (Strongly NP-hard) Clique problem (Strongly NP-hard) Partition problem (NP-hard in the ordinary sence) Vertex Cover (Strongly NP-hard)
PMS	Clique problem (if $p_j = 1$) Partition problem (if $m = 2$)
LSPP	Bin-packing problem (if $p_j = 1$) Partition problem (if $q_j = 1$)
UPT	Bin-packing problem Clique problem (if $q_j = 1$)

7 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 $p_j = \varepsilon,\ q_j = n$, 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 $i \to k \to j$, 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 $1|p-batch, chains, b = \infty|C_{max}$ which is as follows.

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

Let a set $N = \{1, 2, \ldots, n\}$ 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 $i \to j$ 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 $i \to j$, 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 [17], it has been shown that even for the problem $1|p-batch, chains, b = \infty|C_{max}$ with chains as precedence relations, this problem is NP-hard in the strong sense by a reduction from the graph problem *Vertex cover*.

Theorem 7 *The batch-case of the RCPSP is NP-hard in the strong sense.*

Proof.

Assume that we have an instance of problem $1|p-batch, prec, b = \infty|C_{max}$. We construct an instance of the batch-case as follows. All jobs $1, 2, \ldots, n$ from the initial instance correspond to jobs $1, 2, \ldots, n$

of the instance of the RCPSP. The processing times are the same and we have $q_j = 1$ for $j = 1, 2, \ldots, n$. 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 $i \rightarrow k_{ij} \rightarrow j,\ i, j \in N_{long}$. Moreover, let $Q = n$.

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

□

Conclusions

In this paper, we have proven that there exist instances for which the lower bound of Mingozzi has a bad relative error of at least $O(\log n)$ and the calculation of the bound is an NP-hard problem. Moreover, there exist 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 problem $1|p - batch, b = \infty|C_{max}$, 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

[1] Brucker P., Knust S., *Complex Scheduling*, Springer-Verlag Berlin, Heidelberg, Germany, 2006.

[2] Kolisch R., Padman R., *An Integrated Survey of Project Scheduling*, 1997.

[3] Kolisch R., Sprecher A., *PSPLIB - A Project Scheduling Problem Library*, Manuskripte aus den Instituten für Betriebswirtschaftslehre, No. 396, 1996, Kiel, Germany.

[4] Mingozzi A., Maniezzo V., Ricciardelli S., Bianco L., *An Exact Algorithm for Project Scheduling with Resource Constraints Based on New Mathematical Formulation*, Management Science, Vol. 44, 1998, 714 – 729.

[5] Hartmann S., Kolisch R., *Experimental Evaluation of State-of-the-Art Heuristics for the Resource-Constrained Project Scheduling*

Problem, European Journal of Operational Research, Vol. 127, 2000, 394 – 407.

[6] Kolisch R., Hartmann S., *Heuristic algorithms for solving the resource-constrained project scheduling problem - Classification and computational analysis*, Weglarz, J. (Hrsg.): Project Scheduling: Recent Models, Algorithms and Applications, Kluwer, Boston, 1998, 147-178.

[7] Kolisch R., Hartmann S., *Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling: An Update.*, European Journal of Operational Research, Vol. 174(1), 2006, 23-37.

[8] Demeulemeester E.L., Herroelen W.S. , *An Efficient Optimal Solution Procedure for the Preemptive Resource-Constrained Project Scheduling Problem*, European Journal of Operational Research, Vol. 90, 1996, 334 – 348.

[9] Lawler E.L., Lenstra J.K., Rinnooy Kan A. H.G., Shmoys D.B., *Sequencing and Scheduling: Algorithms and Complexity*, Report BS-R8909, Centre for Mathematics and Computer Science, 1989, Amsterdam.

[10] Martello S., Monaci M., Vigo D., *An Exact Approach to the Strip-Packing Problem*, INFORMS Journal on Computing, Vol. 15 (3), 2003, 310 – 319.

[11] Steinberg A., *A Strip-Packing Algorithm with Absolute Performance Bound 2*, SIAM Journal on Computing, Vol. 26 (2), 1997, 401 – 409.

[12] Rykov I., *Approximate Solving of RCPSP*. Abstract Guide of OR 2006, Karlsruhe, Germany, 6.9. - 8.9.2006, p. 226.

[13] Lazarev A.A. and Gafarov E.R., *On Project Scheduling Problem*, Automation and Remote Control, Vol. 69, No. 12, 2008, 2070 – 2087.

[14] Uetz M., *Algorithms for Deterministic and Stochastic Scheduling*, Ph.D. thesis, Cuvillier Verlag, 2002.

[15] Bianco L., Caramia M., *A New Lower Bound for the Resource-Constrained Project Scheduling Problem with Generalized Precedence Relations*, Computers and Operations Research (in print).

[16] Carlier J., Neron E., *On Linear Lower Bounds for the Resource-Constrained Project Scheduling Problem*, European Journal of Operational Research, Vol. 149, 2003, 314 – 324.

[17] T.C.E. Cheng, C.T. Ng, J.J. Yuan, Z.H. Liu, *Single Machine Parallel Batch Scheduling Subject to Precedence Constraints*, Naval Research Logistics, Vol. 57, Issue 7, 2004, 314 – 324.