ISSN 0005-1179, Automation and Remote Control, 2008, Vol. 69, No. 12, pp. 2070–2087. © Pleiades Publishing, Ltd., 2008.
Original Russian Text © A.A. Lazarev, E.R. Gafarov, 2008, published in Avtomatika i Telemekhanika, 2008, No. 12, pp. 86–104.

DISCRETE SYSTEMS

On Project Scheduling Problem[1]

A. A. Lazarev and E. R. Gafarov

Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia

Received October 4, 2007

Abstract—Consideration was given to the resource-constrained project scheduling problem and its special cases. The existing lower estimates of the objective function—minimization of the project time—were compared. It was hypothesized that the optimal value of the objective function of the nonpreemptive resource-constrained project scheduling problem is at most twice as great as that of the objective function with preemption. The hypothesis was proved for the cases of parallel machines and no precedence relation.

PACS numbers: 07.05.Kf, 02.10.Ox

DOI: 10.1134/S0005117908120060

1. INTRODUCTION

The present paper is devoted to speed-oriented scheduling (minimization of the total time of servicing the entire set of demands) under resource constraints and precedence relations. Problems of this sort are frequently encountered in practice. For example, various stages of construction of one or another object require different labor resources, building machinery, materials, and so on. There exist technology-defined precedence relations between individual stages of construction. It is required to construct the schedule so as to meet the precedence relations and resource constraints and, at the same time, to minimize the time of construction.

The resource-constrained project scheduling problem (*RCPSP*) is formulated in Section 2. The existing lower estimates for *RCPSP* and a new lower estimate are given in Section 3. The present authors hypothesized that the values of the objective function of the speed *RCPSP* with and without preemption differ at most by the factor of two. Section 4 outlines a proof of the hypothesis and discusses its practical application and a special case of the problem with one cumulative resource and empty set of the precedence relations. The hypothesis is proved in Section 5 for a special case of parallel machine scheduling. Section 6 reviews the results obtained.

2. FORMULATION OF THE PROBLEM

Given is the set of demands $N = \{1, \ldots, n\}$ and K renewable resources. At each time instant t, Q_k units of the resource k, $k = 1, \ldots, K$, are available. The lengths of servicing $p_i \in \mathbb{Z}^+$ each demand $i = 1, \ldots, n$ are given as well. Servicing of the demand i requires $q_{ik} \leqslant Q_k$ units of the resource $k = 1, \ldots, K$. Upon completion of servicing the demand, the entire released resources can be instantaneously used to service other demands.

For some pairs of demands, given are the precedence constraints $i \to j$ meaning that servicing of the demand j begins only after completing the demand i.

[1] This work was done within the framework of the Grant for Support of the Leading Scientific Schools, project no. NSH-5833.2006.1, and the Foundation for Support of the National Science.

Demand servicing begins at the time instant $t = 0$. Interrupts in servicing of demands are forbidden.

Needed is to determine the time instants of starting servicing of the demand S_i, $i = 1, \ldots, n$, so as to minimize the entire project implementation time $C_{\max} = \max\limits_{i=1,\ldots,n} \{C_i\}$, $C_i = S_i + p_i$. At that, the following constraints must be observed:

—at each time instant $t \in [0, C_{\max})$ and $\sum\limits_{i=1}^{n} q_{ik}\varphi_i(t) \leqslant Q_k$, $k = 1, \ldots, K$, where $\varphi_i(t) = 1$, if the demand i is serviced at time t and $\varphi_i(t) = 0$, otherwise, that is, in the course of execution the demands must be fully provided with resources;

—the precedence relation are not violated, that is, $S_i + p_i \leqslant S_j$ is satisfied if $i \to j$, $i, j \in N$.

This problem is called the *resource-constrained project scheduling problem* (*RCPSP*). The *NP*-hard multidimensional knapsack problem may be reduced to this problem in a polynomial time.

Integrated reviews of the most significant results on *RCPSP*, for example [1], are regularly published by various researchers. The *PSPLIB* library of benchmarks [2] was created for testing and comparing numerous algorithms to solve *RCPSP*. For the time being, the branch-and-bound algorithm of Brucker and others [3] is regarded as the "fastest" one. The algorithm described in [4] deserves special mentioning among the metaheuristic algorithms. Experimental study of some heuristic algorithms can be found in [5] and [6]. An exact algorithm to solve preemptive *RCPSP* was suggested in [7].

Solution of *RCPSP* is representable as a set of the starting instants of demand servicing $S = (S_1, \ldots, S_n)$. A solution meeting the resource and precedence constraints will be called the admissible solution.

The project structure is representable as *demands-in-vertices* of the oriented graph $G = (V, A)$ where some demand from the set $N = \{1, \ldots, n\}$ corresponds to each vertex from $V = \{1, \ldots, n\}$ and the set of arcs $A = \{(i, j) \mid i, j \in V : i \to j\}$ corresponds to the precedence constraints. Obviously, an admissible solution exists only if the precedence graph is acyclic.

Usually, consideration is given to two dummy demands 0 and $n + 1$ with servicing durations $p_0 = p_{n+1} = 0$. The precedence relations are as follows: $0 \to j \to n + 1$, $j = 1, \ldots, n$, $q_{0k} = q_{n+1k} = 0$, $k = 1, \ldots, K$.

We denote by UB the upper boundary of the optimal value of $C_{\max}$. For example, we may assume that $UB = \sum\limits_{i=1}^{n} p_i$.

For each demand $i \in N$, we determine the time window $[r_i, d_i]$ where it must be serviced under any permissible schedule S:

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

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

2.1. Algorithm of RCPSP Scheduling

We present a popular project scheduling algorithm [3].

Algorithm 1 (List Scheduling (*LS*) Algorithm).

(1) Let EL be the list of all demands without predecessors.

$$Q_k(\tau) = Q_k \quad \forall\tau, \quad k = 1, \ldots, K;$$

(2) If $EL = \emptyset$, go to Step 10;

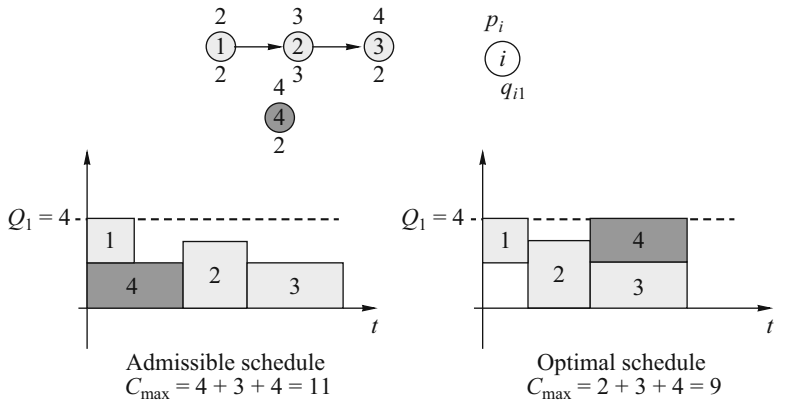

Fig. 1. Example of *RCPSP*.

(3) Take demand $j \in EL$;

(4) $t := \max\limits_{i|(i,j)\in A}\{S_i + p_i\}$. If no predecessors are defined for the demand j, then $S_j = 0$;

(5) If for some $\tau \in [t, t + p_j)$ there exists a resource k such that $q_{jk} > Q_k(\tau)$, then calculate the minimum $t_k > t$ such that the job j can be executed over the interval $[t_k, t_k + p_j)$ if only the resource k is considered. If there is no such resource k, then go to Step 7;

(6) Assume that $t := t_k$ and go to Step 5;

(7) Fix execution of the demand j at the interval $[S_j, C_j) = [t, t + p_j)$;

(8) Reserve resources for the demand $Q_k(\tau) = Q_k(\tau) - q_{jk}$, $k = 1, \ldots, r$, $\tau \in [t, t + p_j)$;

(9) $EL = EL \setminus \{j\}$. Add to EL the successors of the demand j for which all predecessors are arranged and go to Step 2;

(10) End.

The solution obtained depends on the choice of the demand j at Step 3. The concept of the algorithm lies in putting the demand j on the schedule at the earliest time instant where the resource and precedence constraints are not violated. Laboriousness of the algorithm is $O(n^2K)$ operations.

By the *active* schedule is meant an admissible schedule (solution) $S = (S_1, \ldots, S_n)$ for which there is no other admissible schedule $S' = (S'_1, \ldots, S'_n)$ such that at least one of the inequalities $S'_j \leqslant S_j$ and $\forall j \in N$ is nonstrict. The LS algorithm constructs only the active schedules. The optimal schedule, obviously, must be sought among the set of the active schedules.

The following theorem proves that some permutation among the elements of the set N denoted by $\pi = (j_1, \ldots, j_n)$ corresponds to the active schedule $(S_1, \ldots, S_n)$.

Theorem 1 [3]. *The active schedule is uniquely representable as the permutation $\pi = (j_1, \ldots, j_n)$ among n demands.*

The permutation π defines the succession of taking the demand j at Step 3 of the algorithm. Such algorithm is often called the "competition" because at Step 3 the demand can be selected using a certain "priority."

Example 1. Let us consider the example depicted in Fig. 1. The project consists of four demands. On the activity network the duration of demand servicing p_i is shown for the ith demand above the nodes, and the amount q_{i1} of the renewable resource 1 required to service the demand is shown below. Four units of the resource 1 (for example, four installers) are available altogether.

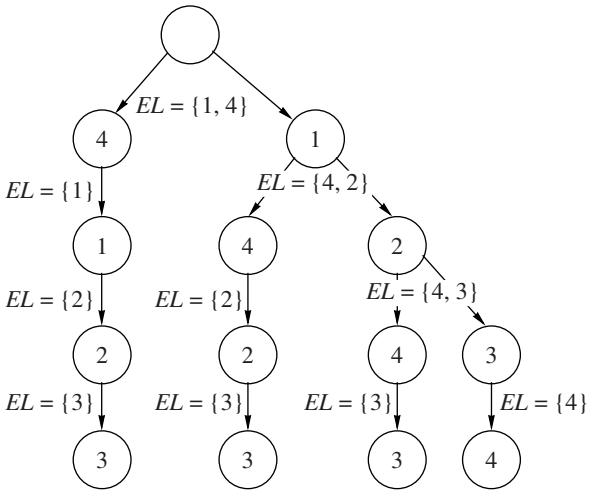

Fig. 2. Tree of seeking the optimal schedule.

Two admissible schedules for which the precedence relations, resource constraints, and demand servicing duration are satisfied are shown in Fig. 1.

We illustrate the algorithm by way of example.

Example. *Step 0.* $EL = \{1, 4\}$;

Steps 3–9. Let $j = 1$. The earliest time of servicing the demand 1: $[0, 2)$. $EL = \{4, 2\}$;

Steps 3–9. Let $j = 4$. The earliest time of servicing the demand 4: $[0, 4)$. $EL = \{2\}$;

Steps 3–9. $j = 2$. The earliest time of servicing the demand 3: $[4, 7)$ because over the interval $[2, 4)$ resources for demand 2 are insufficient (reserved for demand 4). $EL = \{3\}$;

Steps 3–9. $j = 3$. The earliest time of servicing the demand 3: $[7, 11)$ because servicing of demand 2 is completed by the time instant 7. $EL = \emptyset$.

As the result, we get a schedule with $C_{\max} = 11$. The succession of putting the demands on the schedule is representable as the permutation $\pi = (1, 4, 2, 3)$.

If the demands were arranged as $(1, 2, 4, 3)$ or $(1, 2, 3, 4)$, then we would obtain a schedule with $C_{\max} = 9$, that is, the schedule corresponding to the permutation $\pi = (1, 2, 4, 3)$ is superior in terms of the objective function.

An exact branch-and-bound algorithm may be constructed on the basis of the given scheduling algorithm. In it branching occurs at choosing the demand j. For the example of Fig. 1, the scheme of algorithm branching is depicted in Fig. 2. Since the branch-and-bound algorithm is the most efficient tool for solving the problem at hand, determination of the "good" lower estimates becomes urgent.

2.2. RCPSP with Demand Servicing Preemption

The *RCPS Problem* where preemption in demand servicing is permitted, is called the preemption resource-constrained project scheduling problem ($PRCPSP$). Its solution is representable as a collection of sets consisting of pairs of numbers $S' = \{S_1 = \{[s_{11}, c_{11}), \ldots, [s_{1m_1}, c_{1m_1})\}, S_2 = \{[s_{21}, c_{21}), \ldots, [s_{2m_2}, c_{2m_2})\}, \ldots\}$. The demand 1 is continuously serviced over the intervals $[s_{11}, c_{11}), \ldots, [s_{1m_1}, c_{1m_1})$ with $m_1 - 1$ interrupts.

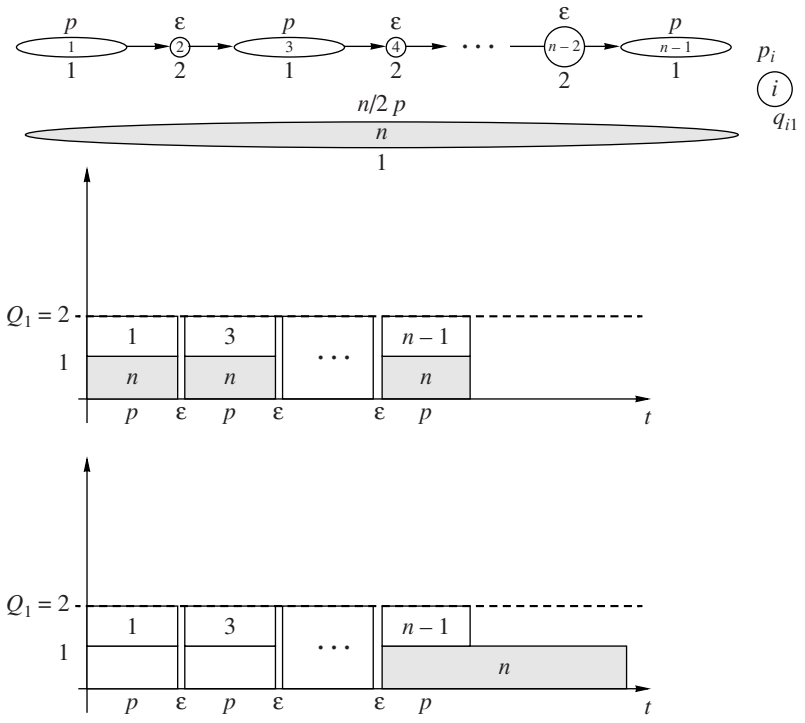

Fig. 3. Example of *RCPSP* with interrupts.

Let $C_{\max}(S^*)$ be the optimal value of the objective function for the example of *RCPSP*. For the same example, we denote by $C_{\max}(S^*_{pmtn})$ the optimal value of the objective function for the case of allowable interrupts.

An example of *RCPSP* and the optimal solutions for the cases with and without interrupts where at each time instant n demands are serviced and two units of resource are available is shown in Fig. 3.

For this example, the project activity network consists of two disjoint fragments. The first fragment represents a succession of "long" and "short" demands (of durations p and ε, respectively, where ε is of a sufficiently small magnitude). The second fragment consists of one demand of duration $(n/2)p$. The objective function is as follows:

$$C_{\max}(S^*) = \left(\frac{n}{2} - 1\right)(p+\varepsilon) + \frac{n}{2}p, \quad C_{\max}(S^*_{pmtn}) = \left(\frac{n}{2} - 1\right)(p+\varepsilon) + p,$$

$$C_{\max}(S^*) - C_{\max}(S^*_{pmtn}) = \left(\frac{n}{2} - 1\right)p < C_{\max}(S^*_{pmtn}),$$

$$C_{\max}(S^*) < 2\,C_{\max}(S^*_{pmtn}),$$

that is, the time of executing a project without interrupts exceeds that of a project with interrupts at most by the factor of two. The estimate is established asymptotically for $n \to \infty$. We draw attention to the fact in the schedule S^*_{pmtn} each fragment $l = [s_{nl}, c_{nl})$ of the demand n is serviced concurrently with another demand.

3. RELATIVE ERROR OF THE *RCPSP* LOWER ESTIMATES

The present section describes the results of analysis of the *RCPSP* lower estimates and, in particular, the relative error of the well-known lower estimates.

3.1. LB_0, Length of the Critical Way

Definition 1. Let the longest path connecting the vertices 0 and $n+1$ on the activity network be called the critical path. Its length is made up of the lengths of servicing the demands in the path.

In the case of no resource constraints, the length of the critical path will be $C_{\max}(S^*)$, that is, the length LB_0 of the critical path is the lower estimate of $C_{\max}(S^*)$.

Assertion 1. *There exists an example of RCPSP for which* $\frac{C_{\max}(S^*)}{LB_0} = n$.

Proof. Let us consider an example of *RCPSP* where the demands $i = 1, \ldots, n$ have servicing durations $p_i = p$. There are no precedence constraints between the demands, and only one resource is required, at that $q_{i1} = Q_1$, $i = 1, \ldots, n$. Obviously, $LB_0 = p$, but $C_{\max}(S^*) = pn$ because no two demands can be serviced in parallel in virtue of the resource constraint $q_{i1} = Q_1$, $i = 1, \ldots, n$. Then, $\frac{C_{\max}(S^*)}{LB_0} = n$.

Since the precedence graph is acyclic, $O(n^2)$ operations are required to determine the estimate LB_0.

3.2. LB_1, Maximum Resource Load

Another lower estimate, LB_1, can be determined in $O(nK)$ operations if each resource is considered separately.

Definition 2. The variable $\sum\limits_{i=1}^{n} q_{ik}p_i$ is called the total load of the resource k.

Obviously, $\sum\limits_{i=1}^{n} q_{ik}p_i \leqslant Q_k C_{\max}(S^*)$, $k = 1, \ldots, K$. Then,

$$LB_1 = \max_{k=1}^{K} \left\lceil \sum_{i=1}^{n} q_{ik}p_i/Q_k \right\rceil$$

is the lower estimate of $C_{\max}(S^*)$.

Assertion 2. *There exists an example of RCPSP for which* $C_{\max}(S^*) - LB_1 = \sum\limits_{i=1}^{n} p_i - 1$.

Proof. Let us consider an example of *RCPSP* where the demands $i = 1, \ldots, n$ have servicing durations p_i. Given are the precedence relations $i \to i+1$, $i = 1, \ldots, n-1$; $q_{i1} = \varepsilon$, $i = 1, \ldots, n$. Let $Q_1 = \sum\limits_{i=1}^{n} p_i\varepsilon$. Obviously, $LB_1 = \sum\limits_{i=1}^{n} q_{i1}p_i/Q_1 = 1$, but $C_{\max}(S^*) = \sum\limits_{i=1}^{n} p_i$. Then, $C_{\max}(S^*) - LB_1 = \sum\limits_{i=1}^{n} p_i - 1$, which proves the assertion.

Obviously, $C_{\max}(S^*) - LB_1 = \sum\limits_{i=1}^{n} p_i$ is satisfied for the example of *RCPSP* for which no resource constraints are given.

3.3. LB_S, Complement of the Critical Path

We denote by CP the set of demands belonging to the critical path and, by e_i the maximum length of the interval in $[r_i, d_i]$ where the demand $i \notin CP$ be can serviced in parallel with the

demands of the critical path without violating the resource constraints. If $e_i < p_i$ is satisfied, then there exists no admissible schedule for which $C_{\max}(S^*) = LB_0$. Then,

$$LB_S = LB_0 + \max_{i \notin CP}\{\max\{p_i - e_i, 0\}\}$$

is also the lower estimate of $C_{\max}(S^*)$.

LB_S can be determined in $O(nK|CP|)$ operations, where $|CP|$ is the number of demands on the critical path.

Assertion 3. *There exists an example of RCPSP for which* $\dfrac{C_{\max}(S^*)}{LB_S} = n/2$.

Proof. Let us consider and example of *RCPSP* where the demands $i = 1, \ldots, n$ have durations of servicing $p_i = p$. There are no precedence constraints on the demands. Let $q_{i1} = Q_1$, $i = 1, \ldots, n$. Then, $LB_0 = p$, and $LB_S = p + \max\{p - 0, 0\} = 2p$ can be readily calculated. However, $C_{\max}(S^*) = pn$ because no two demands can be serviced in parallel by virtue of the resource constraints $q_{i1} = Q_1$, $i = 1, \ldots, n$. Therefore, $\frac{C_{\max}(S^*)}{LB_S} = n/2$, which proves the assertion.

3.4. Lower Mingozzi Estimate

RCPSP as the linear programming problem was formulated in [8]. The lower Mingozzi estimate was obtained through partial relaxation of the original model. A similar method of determining the lower estimate was discussed earlier in [9] and generalized in [10].

Definition 3. If on an oriented graph there exists a path from vertex i to vertex j, the demands i and j will be said to have *indirect* precedence relations.

If $i \to j$, then the given demands will be said to have *direct* precedence relations.

Definition 4. The set of demands $X \subset N$ will be called the *admissible set* if between each pair of demands $i, j \in X$ there are no direct or indirect precedence relations and resource constraints are not violated, $\sum\limits_{i \in X} q_{ik} \leqslant Q_k$, $k = 1, \ldots, K$.

Definition 5. The admissible set X is referred to as *dominating* if there exists no other admissible set Y such that $X \subset Y$ is satisfied.

Let us consider the list of all dominating sets $X_1, \ldots, X_f$ and their corresponding vectors $a^j \in \{0,1\}^n$, $j = 1, \ldots, f$: $a_i^j = 1$ if $i \in X_j$, otherwise, $a_i^j = 0$, $i = 1, \ldots, n$.

Let us determine in terms of x_j the length of the interval where all demands of the set X_j are serviced concurrently. Then, the following linear programming problem enables one to calculate the lower estimate LB_M of $C_{\max}(S^*)$ [8]:

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

This formulation admits that the demand i may be serviced in at least p_i or more time units. In the model, partially violated are the precedence relations, and the demand servicing interrupts are admitted.

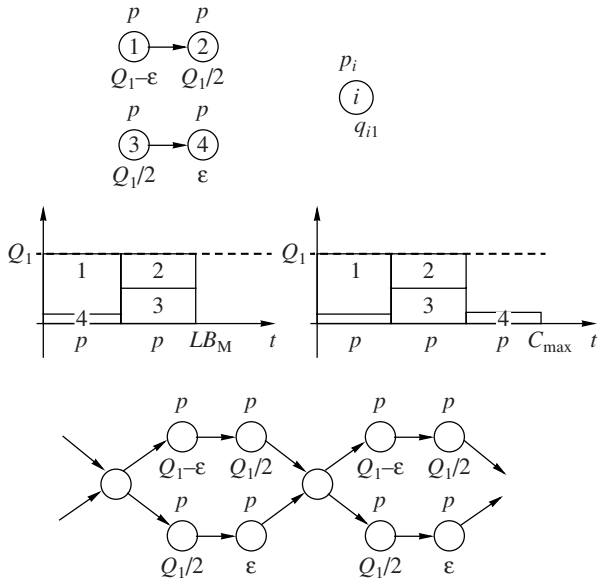

Fig. 4. Illustration to the proof of LB_M.

As was noted in [3], f grows exponentially with n. For $n = 60$ we have $f \approx 300\,000$, and for $n = 90$, even $f = 8\,000\,000$. This paper also described an efficient technique for calculation of the lower estimate LB_M, but the experimental results demonstrated "poor quality of estimation." Nevertheless, for the time being it is one of the "strongest" estimates.

Assertion 4. *Calculation of the estimate LB_M is an NP-hard problem.*

Proof. Let us consider the NP-hard of pallet packing. Given are standard pallets of the lengths of W and n items having each length w_i, $i = 1, \ldots, n$. Needed is to pack all items so as to minimize the number of pallets used.

We rearrange the original packing problem in *RCPSP*. Each demand $i = 1, \ldots, n$ corresponds to the item i. We define $p_i = 1$, $q_{i1} = w_i$, $i = 1, \ldots, n$, $Q_1 = W$. The precedence relations between the demands are not given. Then, $C_{\max}(S^*)$ corresponds to the minimal possible number of pallets in the original problem.

Obviously, $LB_M = C_{\max}(S^*)$ for the constructed example of *RCPSP*. Consequently, the problem of determining the estimate LB_M amounts to the NP-hard problem of pallet packing, which proves the assertion.

As the result of calculation of the estimate LB_M, servicing of some demands can be interrupted. Then, the following assertion is true.

Assertion 5. *There exists an example of RCPSP for which $\frac{C_{\max}(S^*)}{LB_M} \approx 2$.*

Proof. Let us consider the example of Fig. 3 for which the dominating sets $X_1 = \{1, n\}$, $X_2 = \{2\}$, $X_3 = \{3, n\}$, $X_4 = \{4\}, \ldots, X_{n-1} = \{n-1, n\}$ and the optimal solution of the linear programming problem (1) $x_1 = x_3 = \ldots = x_{n-1} = p$, $x_2 = x_4 = \ldots = x_{n-2} = \varepsilon$ are obtained at calculating LB_M.

The values of n, p, and ε can be selected so as to obtain $\frac{C_{\max}(S^*)}{LB_M} \approx 2$, which proves the assertion.

Some precedence relations can be violated as the result of calculating the estimate LB_M. Then, the following assertion is valid.

Assertion 6. *There exists an example of RCPSP for which $\frac{C_{\max}(S^*)}{LB_M} = 1.5$ and the precedence relations are violated "in estimate."*

Proof. Let us consider the example of Fig. 4 for $Q_1/2 \gg \varepsilon$. Calculation of the estimate LB_M provides $X_1 = \{1, 4\}$, $X_2 = \{2, 3\}$, $x_1 = p$, $x_2 = p$, $LB_M = 2p$, but $C_{\max}(S^*) = 3p$. Then, $\frac{C_{\max}(S^*)}{LB_M} = 1.5$, which proves the assertion.

A modified Mingozzi estimate LB_B with allowance for $[r_i, d_i]$, $i = 1, \ldots, n$, was presented in [3]. The assertions and proofs for this estimate are the same.

3.5. Estimate LB_{LG}

Let us strengthen the estimate LB_1 by considering also $[r_i, d_i]$, $i = 1, \ldots, n$, and the *maximum feasible load level* at each point t.

The algorithms for the Partition problem are used to determine the estimate LB_{LG}.

Problem 1 (Partition problem). Given is an ordered set $B = \{b_1, b_2, \ldots, b_n\}$ of n positive numbers. Needed is to decompose B into two subsets B_1 and B_2, $B_1 \bigcap B_2 = \emptyset$ and $B_1 \bigcup B_2 = B$, so as to minimize

$$\left| \sum_{b_i \in B_1} b_i - \sum_{b_i \in B_2} b_i \right| \longrightarrow \min .$$

We describe a *modified* Partition problem which is required to calculate LB_{LG}.

Problem 2. Given is an ordered set of n positive integers $B = \{b_1, b_2, \ldots, b_n\}$ and the number $A \leqslant \sum\limits_{b_i \in B} b_i$. Needed is to specify a subsets of the numbers $B_1 \in B$ so as to minimize the value :

$$\left| A - \sum_{b_i \in B_1} b_i \right| \longrightarrow \min .$$

The following assertion demonstrates that the modified Partition problem comes to the the Partition problem.

Assertion 7. *If for the modified Partition problem there exists $B_1 \in B$ such that $\sum\limits_{b_i \in B_1} b_i = A$, then for the Partition problem with the set of numbers $\overline{B} = B \cup \{b_{n+1}, b_{n+2}\}$, where $b_{n+1} = A + \sum\limits_{b_i \in B} b_i$ and $b_{n+2} = 2 \sum\limits_{b_i \in B} b_i - A$, there exist two subsets B_1 and B_2, $B_1 \bigcap B_2 = \emptyset$ and $B_1 \bigcup B_2 = \overline{B}$, $\sum\limits_{b_i \in B_1} b_i = \sum\limits_{b_i \in B_2} b_i$.*

Proof. To reduce the modified problem to the original one, we complement the set B by two demands $b_{n+1} = A + \sum\limits_{b_i \in B} b_i$ and $b_{n+2} = 2 \sum\limits_{b_i \in B} b_i - A$ and consider the Partition problem with the set of numbers $B \cup \{b_{n+1}, b_{n+2}\}$.

Obviously, the numbers b_{n+1} and b_{n+2} will be in different subsets B_1 and B_2. Let us specify a subset of numbers $B_1 \in B$ so that $A = \sum\limits_{b_i \in B_1} b_i$ and consider two sets

$$\overline{B_1} = \{b_{n+2}\} \cup B_1 \quad \text{and} \quad \overline{B_2} = \{b_{n+1}\} \cup B \setminus B_1 .$$

Then,

$$\sum_{b_i \in \overline{B_1}} b_i = A + 2 \sum_{b_i \in B} b_i - A = 2 \sum_{b_i \in B} b_i$$

and

$$\sum_{b_i\in\overline{B_2}} b_i = A + \sum_{b_i\in B} b_i + \sum_{b_i\in B} b_i - A = 2\sum_{b_i\in B} b_i,$$

that is, $\sum\limits_{b_i\in\overline{B_1}} b_i = \sum\limits_{b_i\in\overline{B_2}} b_i$, which proves the assertion.

Obviously, the Partition problem may be reduced to the modified Partition problem by assuming that $A = \frac{1}{2}\sum\limits_{b_i\in B} b_i$, that is, both problems may be regarded as equivalent.

For both Partition problems, an efficient graphic algorithm which in practice showed itself to good advantage in comparison with the existing dynamic programming algorithms [12] is given in [11]. This graphic algorithm allows one to handle examples with noninteger values of b_i and A.

We denote by $z_0 < z_1 < \ldots < z_\tau$ the ordered list of different values of r_i, d_i. For each interval, $I_t = [z_t, z_{t-1})$, $t = 1, \ldots, \tau$, we determine the set Y_t of demands for which $r_i \leqslant z_{t-1} \leqslant z_t \leqslant d_i$.

The estimate LB_{LG} is calculated independently for each resource $k = 1, \ldots, K$.

Algorithm to calculate LB_{LG}.

Algorithm 2. (1) $G := 0$; $LB_{LG} := 0$; $t := 1$;

(2) Take into consideration the volume of work for the resource k for the demands accessible for servicing from the time instant z_{t-1}. $G := G + \sum\limits_{i|z_{t-1}=r_i} q_{ik}p_i$;

(3) Construct an example of the modified Partition problem: $B := \{q_{ik}|i \in Y_t\}$, $A := Q_k$;

(4) Solve the example of the modified Partition problem and determine the subset B' and $H = \sum\limits_{b_i\in B'} b_i$, $A - \sum\limits_{b_i\in B'} b_i \geqslant 0$. The value of H is the *maximum possible load level*;

(5) Calculate the length of the rectangle of height H and volume at most G. $l := \min\{z_t - z_{t-1}, G/H\}$;

(6) $G := G - lH$, $LB_{LG} := z_t + l$;

(7) If $t = \tau$, the go to Step 8, otherwise, $t := t + 1$, and go to Step 2;

(8) End.

Assertion 8. *The value of LB_{LG} established by Algorithm 2 is the lower estimate of $C_{\max}(S^*)$.*

The proof follows from the algorithm to calculate LB_{LG}.

Assertion 9. *Satisfied is $LB_{LG} \geqslant LB_0 - p_j$, where j is the last demand on the critical path.*

Proof. We assume without loss of generality that $LB_0 = d_{\max}$. Then, $LB_0 = d_{\max} = \max\limits_{j\in N} d_j = z_\tau$. At that, $d_{\max} = z_\tau$. We obtain from the algorithm that $LB_{LG} \geqslant z_{\tau-1} = z_\tau - p_j$, where j is the last demand on the critical path. Then, $LB_{LG} \geqslant LB_0 - p_j$, which proves the assertion.

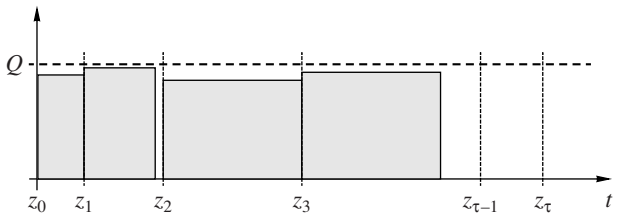

Fig. 5. Estimate LB_{LG}.

In contrast to the estimate LB_M, the given estimate can be modified to calculate the lower estimate of problems where Q_k depends on time t (there exist time intervals over which $Q_k = Q_k(t)$), which is useful in practice and for implementation of some branch-and-bound algorithms.

For the case where Q_k is constant, that is, independent of time t, the estimate LB_{LG} is calculated at most in $O\left(n^2 \sum\limits_{i=1}^{n} q_{ik}\right)$ operations. The experiments showed that for the vast majority of examples the algorithms described in [11] solve the Partition problem in time $O(n^2)$. The estimate LB_{LG} can be calculated for the majority of example in time $O(n^3)$.

3.6. Estimate LB_{SPP}

It seems only logical to use the solution of the *strip packing problem* (SPP) as the lower estimate of *RCPSP*.

Strip packing. Given are a strip of height R and n items to be packed on it without intersections so as to minimize the occupied strip width. For each item $i = 1, \ldots, n$, its height r_i and length p_i are defined. It is forbidden to turn and tear the items. The optimal length of the strip is denoted by SPP^*.

An example of *RCPSP* for which defined are one power resource $Q_1 = R$ and n demands of durations p_i and resource needs $q_{i1} = r_i$, $i = 1, \ldots, n$, "corresponds" to each example of SPP. The precedence relations between the demands are not defined.

As was shown in [13], there are examples for which $SPP^* \geqslant C_{\max}(S^*)$ for the corresponding example of *RCPSP*.

As was proved above, all available estimates (LB_0, LB_1, LB_s) are either ineffective or determination of their (LB_{LG}, LB_M) generally is an NP-hard problem. Lack of efficient methods for determination of the lower estimates prevents practical use of the branch-and-bound, branch & cuts, and other methods.

4. RELATION OF THE OPTIMAL VALUES OF THE OBJECTIVE FUNCTION FOR THE PROBLEMS WITH AND WITHOUT INTERRUPTS

Lemma 1. *For any example of $PRCPSP$, there exists an optimal solution under which at each time instant $t \in [s_{il}, c_{il})$ other demands, except, possibly, for the last fragment, are serviced concurrently with each fragment $l = [s_{il}, c_{il})$ of the "interrupted" demand i.*

Proof. Let us assume that in the optimal solution no demand is serviced in parallel with a fragment l of the "interrupted" demand i. We transform the solution by "shifting" the fragment l to the next fragment $l + 1$ (or the preceding fragment $l - 1$) of the demand i (see Figs. 6b and 6c). At that, the demands serviced "in between" the fragments l and $l + 1$ are shifted by $c_{il} - s_{il}$ to the "left" (correspondingly, "right"). By repeating this operation, we obtain a schedule where the demand i is not interrupted and at that the value of the objective function does not increase. In the course of executing the operation, other "interrupted" demands do not occur.

A situation is possible where not the entire fragment l is shifted but only its part which is not "covered" by other demands. At that, we try to shift this part to the last fragment m_i (see Figs. 6d and 6e).

Lemma 1 is required to support the following hypothesis. In what follows, we consider only the schedules satisfying this lemma.

Hypothesis 1. For any example of RCPSP and the corresponding example of $PRCPSP$, $C_{\max}(S^*) \leqslant 2C_{\max}(S')$ is satisfied.

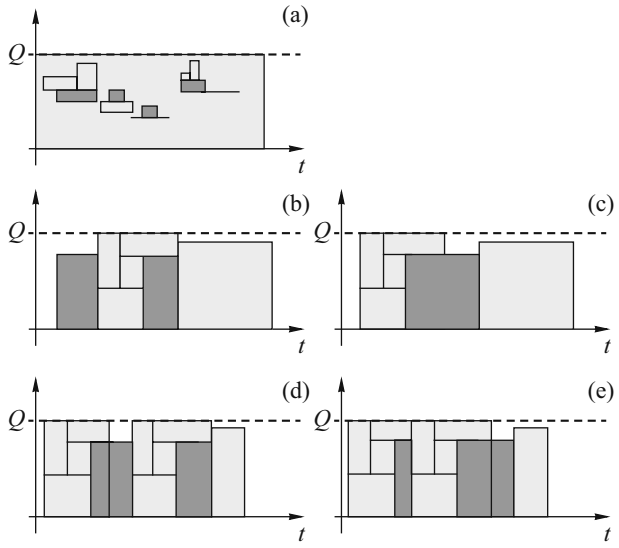

Fig. 6. Illustration to Lemma 1.

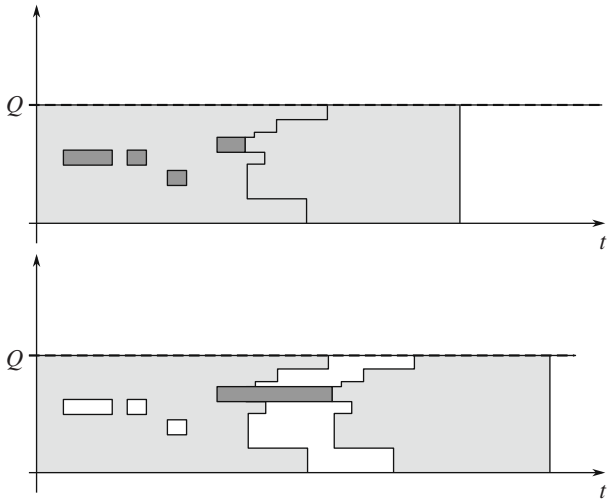

Fig. 7. One shift.

The present authors hypothesize that the optimal value of the objective function (the total time of project execution) in the problem without demand interruption is at most twice the value of $C_{\max}$ in the corresponding problem with permitted interrupts.

Outline of the proof. Let there be the optimal solution S' and the value of $C_{\max}(S')$ for the example of $PRCPSP$ (with interrupts). We rearrange the solution S' in an admissible solution S and the value of $C_{\max}(S)$ for the corresponding example of $RCPSP$ (without interrupts). The value of $C_{\max}(S)$ is the upper estimate of the optimal $C_{\max}(S^*)$.

We prove that $C_{\max}(S) \leqslant 2C_{\max}(S')$. Then, $C_{\max}(S^*) \leqslant C_{\max}(S) \leqslant 2C_{\max}(S')$, and the hypothesis is proved.

The solution S' is rearranged in the admissible solution S as follows.

Single shift. Let in the schedule S' servicing of only one demand i be interrupted.

We fix servicing of this demand "entirely" to the interval $[s_{im_i}, s_{im_i} + p_i)$. At that, all demands j for which $s_{j1} \leqslant c_{im_i}$ "remain" in their places. The rest of demands are shifted to the "right" by $p_i - (c_{im_i} - s_{im_i})$ (see Fig. 7).

As the result of this operation, the value of $C_{\max}$ was increased by $p_i - (c_{im_i} - s_{im_i})$, that is, less than twice.

In virtue of Lemma 1, the duration of servicing the "nonshifted" fragment exceeds p_i.

The case where under the schedule S' servicing of only two demands (*two shifts*) is interrupted is proved along the same lines.

Three and more shifts were not considered by the present authors yet.

Proof or refutation of the hypothesis under consideration will provide the following results:

If the hypothesis is true, then solved a problem with interruptions, we shall receive the bottom estimate $LB = C_{\max}(S')$ and a unattainable top estimate $UB = 2C_{\max}(S')$ for a problem *RCPSP* without interrupts. The top estimate can be used in a branches and bounds method, effectively cutting bad decisions. To the contrary, having received an estimate $C_{\max}(S^*)$ for a problem *RCPSP*, we shall find the top estimate for a problem *PRCPSP*;

If the hypothesis will appear is incorrect and it will turn out $C_{\max}(S^*) \approx O(n)C_{\max}(S')$ (i.e. a difference by way of n times), the best known estimate LB_M, used in a branches and bounds method, can deviate from $C_{\max}(S^*)$ in $O(n)$ times. In this case use of a method of type of branches and bounds method with the bottom estimation LB_M will appear inefficient generally.

4.1. Proof of the Hypothesis for the Case of RCPSP with One Resource

Consideration is given to a special NP-hard case of *RCPSP* with one cumulative resource of power Q_1 and empty graph of precedence relations. For $q_{j1} = 1$ and $Q_1 = m$, we have the classical problem of m-decomposition [14]. One can see some resemblance of this special case to the strip packing problem (SPP). Nevertheless, there exist substantial distinctions (see Section 3.6). For example, at visual representation of *RCPSP*, the "rectangles" corresponding to the demands can be torn vertically, which in inadmissible in SPP.

Let us consider a scheduling algorithm LS with the following domination rule: from the set EL of the not yet scheduled demands that one is taken whose servicing can be started before other demands from EL without violating the resource constraints. Stated differently, at Step 2 of the algorithm LS each demand $j \in EL$ is assigned the "possible time of starting" r_j beginning from which the demand can be serviced without violating the resource constraints. Selected is the demand $j \in EL$ with the least r_j. The scheduling algorithm with such domination rule will be denoted by LS_t.

Example 2. Let us consider an example (only without the precedence relations) familiar to the present authors (Fig. 8). The algorithm LS_t is illustrated by way of example.

Step 1. $EL = \{1, 2, 3, 4\}$;

Steps 3–9. $r_1 = r_2 = r_3 = r_4 = 0$. Let $j = 1$. The earliest time of execution of demand 1: $[0, 2)$. $EL = \{2, 3, 4\}$;

Steps 3–9. $r_3 = r_4 = 0$, $r_2 = 2$. Let $j = 4$. The earliest time of execution of demand 4: $[0, 4)$. $EL = \{2, 3\}$;

Steps 3–9. $r_3 = 0$, $r_2 = 4$. Then, $j = 3$. The earliest time of execution of demand 3: $[2, 6)$. $EL = \{2\}$;

Steps 3–9. $r_2 = 6$, $j = 2$. The earliest time of execution of demand 2: $[6, 9)$. $EL = \emptyset$.

Figure 8 schematizes the resulting schedule and the level of resource load corresponding to it.

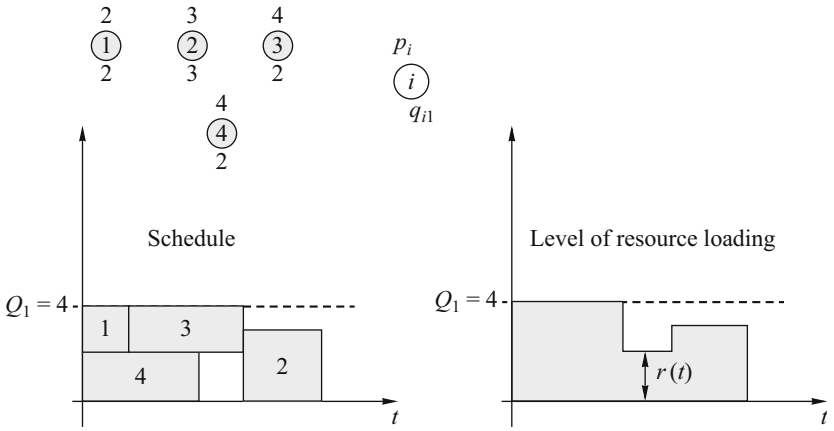

Fig. 8. Example *RCPSP*.

Theorem 2. *$C_{\max}(LS_t) - p_{\max} < 2C^*_{\max}$, where $p_{\max} = \max\limits_{j \in N} p_j$.*

Proof. Let for some example I satisfying the special case under consideration the algorithm LS_t construct an admissible schedule $S = (S_1, S_2, \ldots, S_n)$ and determine the value of the objective function $C_{\max}(LS_t)$.

Let us consider the level of loading of the cumulative resource at each time instant $t \in [0, C_{\max}(LS_t))$ and denote by $N(t) \subseteq N$ the set of demands serviced at the time instant t. We call the value $\rho(t) = \sum\limits_{j \in N(t)} q_{j1}$ the *level of the resource loading* at the time instant t. Let us consider the values of t for which $\rho(t) < Q_1/2$. We demonstrate that over the interval $[0, C_{\max}(LS_t))$ there can be at most two such intervals $[t_1, t_2)$, $[t_3, t_4)$, $0 \leqslant t_1 < t_2 < t_3 < t_4 \leqslant C_{\max}(LS_t)$ over which $\rho(t) < Q_1/2$.

Let us assume that on the contrary there exist three intervals $[t_1, t_2)$, $[t_3, t_4)$, and $[t_5, t_6)$, $0 \leqslant t_1 < t_2 < t_3 < t_4 < t_5 < t_6 \leqslant C_{\max}(LS_t)$, where $\rho(t) < Q_1/2$ is satisfied and consider two cases:

(1) Let the demand $j \in N(t')$ be serviced at some point $t' \in [t_3, t_4)$ with the level of loading $\rho(t') < Q_1/2$. At that, $t_2 \leqslant S_j < t_4$, that is, the demand j is not serviced at any point of the interval $[t_1, t_2)$. Since $\rho(t') < Q_1/2$, we see that $q_{j1} < Q_1/2$. Then, however, according to the algorithm LS_t servicing of the demand j must be started not later than at the instant t_1 because $\rho(t_1) < Q_1/2$. We arrive at contradiction.

Similar reasoning is carried out for the demand $j \in N(t')$, where $t_2 \leqslant S_j < t_6$ at the point $t' \in [t_5, t_6)$ with the loading level $\rho(t') < Q_1/2$;

(2) Let there exist no demand j serviced at some point $t' \in [t_3, t_4)$ for which $t_2 \leqslant S_j < t_4$ and no demand i matrix at some point $t' \in [t_5, t_6)$ for which $t_2 \leqslant S_i < t_6$. Otherwise, see Item 1.

Then, there exists a point $t^* \in [t_1, t_2)$ such that $N(t') = N(t^*) \ \ \forall t' \in [t_3, t_4)$, is satisfied and $N(t'') \subseteq N(t^*)$ for some point $t'' \in [t_5, t_6)$ (see Fig. 9).

Let us consider the demand $j \in N(t')$, $t' \in [t_4, t_5)$, $j \notin N(t^*)$. Such a demand exists because $\rho(t') \geqslant Q_1/2$ by our assumption. Yet, according to the algorithm LS_t servicing of the demand j then must be started not later than at the instant t_3 (see Fig. 9). We arrive at contradiction.

Thus, we proved that over the interval $[0, C_{\max}(LS_t))$ there can be at most two intervals $[t_1, t_2)$, $[t_3, t_4)$, $0 \leqslant t_1 < t_2 < t_3 < t_4 \leqslant C_{\max}(LS_t)$, over which $\rho(t) < Q_1/2$ is satisfied.

Additionally, if there exists a second interval $[t_3, t_4)$, then $\forall t \in [t_3, t_4)$ satisfied is $N(t) \subseteq N(t^*)$ for some point $t^* \in [t_1, t_2)$. In what follows, the interval $[t_1, t_4)$ will be considered "entirely."

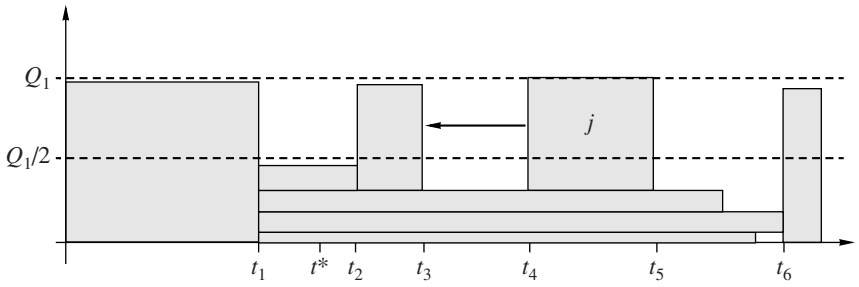

Fig. 9. Drawing 1 for the proof of Theorem 2.

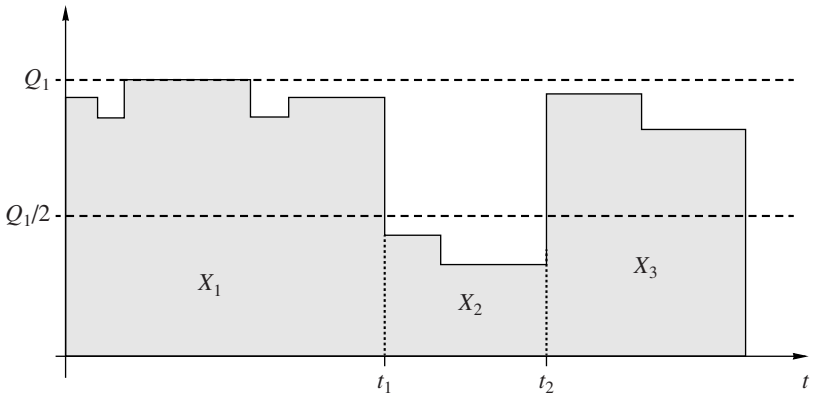

Fig. 10. Drawing 2 for the proof of Theorem 2.

We show that the length of such interval $t_2 - t_1 \leqslant p_{\max}$. We assume that on the contrary $t_2 - t_1 > p_{\max}$. Then, there exists a demand j, $F_j = S_j + p_j = t_2$, for which $S_j = t_2 - p_j \geqslant t_2 - p_{\max} > t_2 - (t_2 - t_1) = t_1$, that is, $S_j > t_1$. Since $\rho(S_j) < Q_1/2$, we get $q_{j1} < Q_1/2$. Yet then by the algorithm LS_t servicing of the demand j must be started not later than at the instant t_1 because $\rho(t_1) < Q_1/2$. We arrive at contradiction. Consequently, the length of the interval $t_2 - t_1 \leqslant p_{\max}$ (is satisfied also for the interval $[t_1, t_4) : t_4 - t_1 \leqslant p_{\max}$).

We represent schematically the resulting load of the equipment as three units X_1, X_2, and X_3 (see Fig. 10). The fragments X_1, X_2, and X_3 "lie" within the intervals $[0, t_1)$, $[t_1, t_2)$, and $[t_2, C_{\max}(LS_t))$, respectively.

The total area of the three figures X_1, X_2, and X_3 will be called the *utilized resource volume* denoted by V. Obviously, $V = \sum\limits_{j:=1}^{n} p_1 q_{j1}$. Correspondingly, by the *nonutilized resource volume* is meant $V_e = Q_1 C_{\max}(LS_t) - V$.

Obviously, the solution determined by the algorithm LS_t can be improved from the standpoint of the objective function if the resources are utilized "more reasonably" without idling, that is, if V_e (and, correspondingly, $C_{\max}$) are reduced.

Let us consider an example I' obtained from the original example I by reducing p_j for dome demands $j \in N$. If $j \in N(t')$ is satisfied for some demand $j \in N$ and some point $t' \in [t_1, t_2)$, then we assume that $p'_j = p_j - (\max(t_1, S_j) - \min(F_j, t_2))$. For the rest of the demands, we assume that $p'_j = p_j$.

Let us consider the schedule $S' = (S'_1, S'_2, \ldots, S'_n)$ for the example I':

$$S'_j = S_j, \quad S_j < t_1;$$
$$S'_j = t_1, \quad S_j \in [t_1, t_2];$$
$$S'_j = S_j - (t_2 - t_1), \quad S_j > t_2,$$

that is, for the schedule S' the fragment X_2 (see Fig. 10) was reduced and the fragment X_3 was shifted to the point t_1.

Obviously, the schedule S' is admissible for the example I' and is constructed by the algorithm LS_t. We denote by $\overline{C}_{\max}(LS_t)$ the value of the objective function for the example I' under the schedule S' and by $\overline{C}^*_{\max}$, the optimal value of the objective function for the example I'. Then, $\overline{C}_{\max}(LS_t) = C_{\max}(LS_t) - (t_2 - t_1)$, $\overline{C}^*_{\max} \leqslant C^*_{\max}$, where $C^*_{\max}$ is the optimal value of the objective function for the original example I.

For the schedule S' $\forall t \in [0, \overline{C}_{\max}(LS_t))$, $\rho(t) > Q_1/2$ is satisfied. Then, $V > V_e$ for the example I', and, consequently, $2\overline{C}^*_{\max} > \overline{C}_{\max}(LS_t)$. We get $2C^*_{\max} \geqslant 2\overline{C}^*_{\max} > \overline{C}_{\max}(LS_t) = C_{\max}(LS_t) - (t_2 - t_1) \geqslant C_{\max}(LS_t) - p_{\max}$ and denote by $C^*_{\max}(pmtn)$ the optimal value of the objective function for the problem with demand servicing interrupts.

Theorem 3. $2C^*_{\max}(pmtn) > C^*_{\max} - p_{\max}$.

Proof. We continue the proof of the above theorem. Since $2C^*_{\max}(pmtn) \geqslant 2\overline{C}^*_{\max}(pmtn) \geqslant \overline{C}_{\max}(LS_t) \geqslant C_{\max}(LS_t) - p_{\max}$, we obtain from that $2C^*_{\max}(pmtn) \geqslant C_{\max}(LS_t) - p_{\max} \geqslant C^*_{\max} - p_{\max}$.

5. SCHEDULING FOR PARALLEL MACHINES

In the present section we consider a special case of *RCPSP* of parallel machine scheduling. Given are n demands, the graph of precedence relations, and the durations p_j of demand servicing. The demands are to be serviced on m identical parallel machines. The objective function $C_{\max}$ is the common instant of completing all demands. For the given special case there exists a single cumulative resource of power $Q_1 = m$, and at that $q_{j1} = 1$, $j = 1, \ldots, n$. We denote the problem of parallel machine scheduling by PMS or $Pm|prec|C_{\max}$ as is customary in the scheduling theory. This problem is NP-hard in the strong sense [15]. Graham [16] demonstrated that the simple scheduling algorithm works rather well here. We denote by $C_{\max}(LS)$ the value of the objective function obtained by the scheduling algorithm (List Scheduling). Then, $C_{\max}(LS)/C^*_{\max} \leqslant 2 - \frac{1}{m}$, where $C^*_{\max}$ is the optimal value of the objective function.

Let us consider the PMS problem with demand servicing interrupts $Pm|prec, pmtn|C_{\max}$ and denote by $C^*_{\max}(pmtn)$ the optimal value of its objective function.

The following fact supports Hypothesis 1 that was put forward by the present authors.

Theorem 4. $C^*_{\max} \leqslant 2C^*_{\max}(pmtn)$.

Proof. As was proved in [16],

$$C_{\max}(LS) \leqslant \frac{\sum p_j}{m} + CP,$$

where CP is the length of the critical path, that is, the upper bound of $C_{\max}(LS)$ for the problem $Pm|prec|C_{\max}$ is smaller than the sum of two simple lower estimates for the same problem.

One can readily show that $C^*_{\max}(pmtn) \geqslant \dfrac{\sum p_j}{m}$ and $C^*_{\max}(pmtn) \geqslant CP$. Consequently,

$$C_{\max}(LS) \leqslant 2C^*_{\max}(pmtn).$$

Therefore, Hypothesis 1 is true for the problem $Pm|prec|C_{\max}$. This result was first obtained by R.L. Graham who, unfortunately, did not publish it [15].

6. CONCLUSIONS

The present paper considered the problem of project scheduling with regard for the resource constraints and its special cases. The objective function lies in minimizing the total project execution time. The existing lower estimates of the objective function were subjected to comparative analysis. Calculation of the most efficient Mingozzi lower estimate was shown to be an NP-hard problem.

It as hypothesized that for the considered problem without demand servicing interrupts the optimal value of the objective function is at most twice as great as the optimal value of the objective function for the corresponding problem with interrupts. Proofs of the hypothesis were given for the cases of the problem with parallel machines and without the precedence relations in the demand servicing. Proofs or disproofs of the hypothesis under consideration will give the following results:

—if the hypothesis is true, then solution of the problem with interrupts will establish the lower estimate $LB = C_{\max}(S')$ and the unattainable upper estimate $UB = 2C_{\max}(S')$ for $RCPSP$ without interrupts. The upper estimate may be used in the branch-and-bound method for effective rejection of the "bad" solutions. The other way round, by determining $C_{\max}(S^*)$ for $RCPSP$ we get the upper estimate for $PRCPSP$;

—if the hypothesis is not true and we get $C_{\max}(S^*) \approx O(n)C_{\max}(S')$, that is, an n-fold difference, then the "best known" lower estimate LB_M used in the branch-and-bound method may deviate from $C_{\max}(S^*)$ by the factor of $O(n)$. In this case, method like branch-and-bound (branch & bounds, Constraint Programming, and so on) with the lower estimate LB_M prove to be ineffective.

REFERENCES

1. Kolish, R. and Padman, R., An Integrated Survey of Project Scheduling, in *Manuscripte aus den Institut fur Betriebswirtschaftslehre*, Kiel, 1997.

2. Kolish, R. and Sprecher, A., PSPLIB—A Project Scheduling Problem Library, in *Manuscripte aus den Institut fur Betriebswirtschaftslehre*, Kiel, 1996, no. 396.

3. Brucker, P. and Knust, S., *Complex Scheduling*, Heidelberg: Springer, 2006.

4. Merkel, D., Middendorf, M., and Schmeck, H., Ant Colony Optimization for Resource-Constrained Project Scheduling, *IEEE Trans. Evolut. Comput.*, 2002, vol. 6, no. 4, pp. 333–346.

5. Hartmann, S. and Kolish, R., Experimental Evaluation of State-of-the-Art Heurisitcs for the Resource-constrained Project Scheduling Problem, *EJOR*, 2000, no. 127, pp. 394–407.

6. Kolish, R. and Hartmann, S., Heuristics Algorithms for Solving the Resource-Constrainde Project Scheduling Problem: Classification and Computational Analysis, in *Manuscripte aus den Institut fur Betriebswirtschaftslehre*, Kiel, 1998, no. 469.

7. Demeulemeester, E.L. and Herroelen, W.S., An Efficient Optimal Solution Procedure for the Preemptive Resource-constrained Project Scheduling Problem, *EJOR*, 1996, no. 90, pp. 334–348.

8. Mingozzi, A., Maniezzo, V., Ricciardelli, S., and Bianco, L., An Exact Algorithm for Project Scheduling with Resource Constraints Based on New Mathematical Formulation, *Manag. Sci.*, 1998, no. 44, pp. 714–729.

9. Burkov, V.N., Problems of Optimum Distribution of Resources, *Control Cybernet.*, 1972, vol. 1, no. 1/2, pp. 27–41.

10. *Matematicheskie osnovy upravleniya proektami* (Mathematical Methods of Project Management), Burkov, V.N., Ed., Moscow: Vysshaya Shkols, 2005.

11. Lazarev, A.A., A Graphical Approach to the Problems of Combinatorial Optimization, *Avtom. Telemekh.*, 2007, no. 4, pp. 13–23.

12. Kellerer, H., Pferschy, U., and Pisinger, U., *Knapsack Problems*, Berlin: Springer, 2004.

13. Rykov, I., Approximate Solving of *RCPSP*, in *Abstract Guide of OR 2006*, Karlsruhe, p. 226.

14. Garey, M.L. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco: Freeman, 1979. Translated under the title *Vychislitel'nye machiny i trudnoreshaemye zadachi*, Moscow: Mir, 1982.

15. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., and Schmoys, D.B., Sequencing and Scheduling: Algorithms and Complexity, in *Report BS-R8909*, Centre for Math. Comput. Sci. Amsterdam, 1989.

16. Graham, R.L., Bounds for Certain Multiprocessing Anomalies, *SIAM J. Appl. Math.*, 1966, no. 17, pp. 263–269.

This paper was recommended for publication by D.A. Novikov, a member of the Editorial Board