Chapter

Polynomial Algorithm for Constructing Pareto-Optimal Schedules for Problem $1|r_j|L_{\max}, C_{\max}$

Alexander A. Lazarev and Nikolay Pravdivets

Abstract

In this chapter, we consider the single machine scheduling problem with given release dates, processing times, and due dates with two objective functions. The first one is to minimize the maximum lateness, that is, maximum difference between each job due date and its actual completion time. The second one is to minimize the maximum completion time, that is, to complete all the jobs as soon as possible. The problem is NP-hard in the strong sense. We provide a polynomial time algorithm for constructing a Pareto-optimal set of schedules on criteria of maximum lateness and maximum completion time, that is, problem $1|r_j|L_{max}, C_{max}$, for the subcase of the problem: $d_1 \le d_2 \le \ldots \le d_n; d_1 - r_1 - p_1 \ge d_2 - r_2 - p_2 \ge \ldots \ge d_n - r_n - p_n$.

Keywords: single machine scheduling, two-criteria scheduling, Pareto-set, Pareto-optimality, minimization of maximum lateness, minimization of maximum completion time, polynomial time algorithm

1. Introduction

We consider a classical scheduling problem on a single machine. A release time of each job is predefined and represents the minimum possible start time of the job. When constructing schedules, we consider two objective functions. The first one is to minimize the maximum lateness, that is, maximum difference between each job due date and its actual completion time. The second one is to minimize the maximum completion time, that is, to complete all the jobs as soon as possible. The problem is NP-hard in the strong sense [1]. We provide a polynomial time algorithm for constructing a Pareto-optimal set of schedules on criteria of maximum lateness and maximum completion time, that is, problem $1|r_j|L_{max}, C_{max}$, for the subcase of the problem when due dates are: $d_1 \le d_2 \le \ldots \le d_n$; $d_1 - r_1 - p_1 \ge d_2 - r_2 - p_2 \ge \ldots \ge d_n - r_n - p_n$. Example of a problem case that meets these constraints will be the case when all jobs have the same time for processing before due date.

2. Statement of the problem $1|r_j|L_{\max}, C_{\max}$

We consider single machine scheduling problem, where a set of n jobs $N = \{1, 2, \ldots, n\}$ has to be processed on a single machine. Each job we is numbered, that

is, the entry "job j" is equivalent to the entry "job numbered j." Simultaneous executing of jobs or preemptions of the processing of a job are prohibited. For jobs $j \in N$, value r_j is the minimum possible start time, $p_j \geq 0$ is a processing time of job j and d_j is a due date of job j.

The schedule is represented by a set $\pi = \{s_j | j \in N\}$ of start times of each job. By τ, we denote the permutation of $(j_1, ..., j_n)$ elements of the set N. A set of all different schedules of jobs from the set N is denoted by $\Pi(N)$. Schedule π is called *feasible* if $s_j(\pi) \geq r_j$, $\forall j \in N$. We denote the completion time of job $j \in N$ in schedule π as $C_j(\pi)$. Difference $L_j(\pi) = C_j(\pi) - d_j, j \in N$, denotes lateness of job j in the schedule π. Maximum lateness of jobs of the set N under the schedule π is

$$L_{\max}(\pi) = \max_{j \in N} \{C_j(\pi) - d_j\}. \quad (1)$$

We denote the completion time of all jobs of the set N in schedule π as

$$C_{\max}(\pi) = \max_{j \in N} C_j(\pi).$$

The problem is to find the optimal schedule π^* with the smallest value of the maximum lateness:

$$L^*_{\max} = \min_{\pi \in \Pi(N)} L_{\max}(\pi) = L_{\max}(\pi^*). \quad (2)$$

For any arbitrary set of jobs $M \subseteq N$ we also denote:

$$r_M = \min_{j \in M} r_j, \quad d_M = \max_{j \in M} d_j, \quad p_M = \sum_{j \in M} p_j. \quad (3)$$

In the standard notation of Graham et al. [2], this problem is denoted as $1|r_j|L_{\max}$. Intensive work on the solution of this problem has continued since the early 50s of the 20th century. Lenstra et al. [1] showed that the general case of the problem $1|r_j|L_{\max}$ is *NP*-hard in the strong sense.

Potts [3] introduced an iterative version of extended Jackson rule (IJ) [4] and proved that $\frac{L_{\max}(\pi_{IJ})}{L^*_{\max}} \leq \frac{3}{2}$. Hall and Shmoys [5] modified the iterative version and created an algorithm (MIJ) that guarantees the evaluation $\frac{L_{\max}(\pi_{MIJ})}{L^*_{\max}} \leq \frac{4}{3}$. They also presented two approximation schemes that guarantee finding ε-approximate solution in $O\left(n \log n + n(1/\varepsilon)^{O(1/\varepsilon^2)}\right)$ and $O\left((n/\varepsilon)^{O(1/\varepsilon)}\right)$ operations. Mastrolilli [6] introduced an improved approximation scheme with complexity of $O\left(n + (1/\varepsilon)^{O(1/\varepsilon)}\right)$ operations.

A number of polynomially solvable cases of the problem were found, starting with Jackson's early result [4] for the case $r_j = 0, j \in N$, when the solution is a schedule in which jobs are ordered by nondecreasing due dates (by rule *EDD*). Such a schedule is also be optimal for the case when the release times and due dates are associated ($r_i \leq r_j \Leftrightarrow d_i \leq d_j$, $\forall i, j \in N$).

Schedule is constructed according to the extended Jackson rule (Schrage schedule): on the next place in the schedule we select a released non-ordered job with the minimum due date; if there are no such jobs, then we select the job with the minimum release time among the unscheduled jobs.

If process times of all jobs are equal, the complexity can be reduced to $O(n \log n)$ [7]. Vakhania generalized this result [8] considering the case when the processing times of some jobs are restricted to either p or $2p$. An algorithm with complexity of $O(n^2 \log n \log p)$ was suggested.

A case when job processing times are mutually divisible is considered in [9]. Author suggest a polynomial-time algorithm with a complexity of $O(n^3 \log n \log^2 p_{\max})$ operations for solving this case.

Special cases $1|prec; r_j|C_{\max}$, $1|prec; p_j = p; r_j|L_{\max}$ and $1|prec; r_j; pmtn|L_{\max}$ with precedence constraints for jobs have been addressed in works of Lawler [10], Simons [11], Baker et al. [12]. Hoogeveen [13] proposed a polynomial algorithm for the special case when job parameters satisfy the constraints $d_j - p_j - A \le r_j \le d_j - A$, $\forall j \in N$, for some constant A. A pseudo-polynomial algorithm for the NP-hard case when release times and due dates are in the reversed order ($d_1 \le \ldots \le d_n$ and $r_1 \ge \ldots \ge r_n$) was developed in [14].

We denote by $L_j^A(\pi)$ and $C_j^A(\pi)$ the lateness and completion time of job $j \in N$ in schedule π, for instance, A with job parameters $\left\{r_j^A, p_j^A, d_j^A\right\}, j \in N$. Respectively, $L_{\max}^A(\pi) = \max\limits_{j \in N} L_j^A(\pi)$ is a maximum lateness of the schedule π for instance A.

This paper deals with a problem with two objective functions $L_{\max}$ and $C_{\max}$, which in general case can be referred as $1|r_j|L_{\max}, C_{\max}$. This problem was considered in [15], where authors consider some dominance properties and conditions when the Pareto-optimal set can be formed in polynomial time.

Definition 1.1 For any instance A of the problem, each permutation τ of the jobs of the set N is uniquely defines *early schedule* π_τ^A. In the early schedule, each job $j \in N$ starts immediately after the end of the previous job in the corresponding permutation. If the completion time of the previous job is less than the release time of the current job, then the beginning of the current job is equal to its release time. That is, if $\tau = (j_1, j_2, \ldots, j_n)$, then $\pi_\tau^A = (s_{j_1}, s_{j_2}, \ldots, s_{j_n})$, where

$$s_{j_1} = r_{j_1}^A, s_{j_k} = \max\left\{s_{j_{k-1}} + p_{j_{k-1}}^A, r_{j_k}^A\right\}, \quad k = 2, \ldots, n. \tag{4}$$

Early schedules play an important role in our construction, since it is sufficient to check all early schedules to find the optimal schedule of any problem instance.

By τ^A we denote the optimal permutation and π^A we denote the optimal schedule for instance A. Only early optimal schedules are be considered, that is, $\pi^A = \pi_{\tau^A}^A$.

We denote by $\Pi(N)$ the set of all permutations of jobs of the set N, and by Π_A the set of feasible schedules for instance A.

3. Problem $1|d_i \le d_j, d_i - r_i - p_i \ge d_j - r_j - p_j|L_{\max}, C_{\max}$

This section deals with the problem of constructing a Pareto-optimal set by criteria $C_{\max}$ and $L_{\max}$, that is, problem $1|r_j|L_{\max}, C_{\max}$. We suggest an algorithm for constructing a set of schedules $\Phi(N, t) = \left\{\pi_1', \pi_2', \ldots, \pi_m'\right\}$ for which

$$C_{\max}(\pi_1') < C_{\max}(\pi_2') < \ldots < C_{\max}(\pi_m'), \tag{5}$$

$$L_{\max}(\pi_1') > L_{\max}(\pi_2') > \ldots > L_{\max}(\pi_m'). \tag{6}$$

There is no schedule π such that $C_{\max}(\pi) \le C_{\max}(\pi_i')$ and $L_{\max}(\pi) \le L_{\max}(\pi_i')$, at least one of the inequalities is strict for some i, $i = 1, ..., m$. It is shown that $m \le n$.

3.1 Problem properties

We denote the precedence of the jobs i and j in schedule π as $(i \to j)_\pi$. We also introduce

$$r_j(t) = \max\{r_j, t\}; \tag{7}$$

$$r(N,t) = \min_{j \in N}\{r_j(t)\}. \tag{8}$$

In cases when its obvious how many jobs we mean, we write $r(t)$ instead of $r(N,t)$.

We assume that the job parameters satisfy the following constraints:

$$d_1 \le ... \le d_n, \quad d_1 - r_1 - p_1 \ge ... \ge d_n - r_n - p_n. \tag{9}$$

For example, these constraints correspond to the case when $d_j = r_j + p_j + z$, $j = 1, ..., n$, where z is a constant, that is, when all jobs have the same time for processing before due date. A problem with similar constraints but for a single objective function ($L_{\max}$) is considered in [16].

We assume that $|N| > 1$ and t is the time when the machine is ready. From the set N, we find two jobs $f = f(N,t)$ and $s = s(N,t)$ in the following way:

$$f(N,t) = \arg\min_{j \in N}\{d_j | r_j(t) = r(N,t)\}, \tag{10}$$

$$s(N,t) = \arg\min_{j \in N \setminus \{f\}}\{d_j | r_j(t) = r(N \setminus f, t)\}, \tag{11}$$

where $f = f(N,t)$. If $N = \{i\}$, then we set $f(N,t) = i$, $s(N,t) = 0$, $\forall t$. We also define $d_0 = +\infty, f(\emptyset, t) = 0, s(\emptyset, t) = 0$, $\forall t$. For jobs f and s the following properties are true.

Lemma 1.1 If the jobs of the set N satisfy (4), then for any schedule $\pi \in \Pi(N)$ for all $j \in N \setminus \{f\}$, for which $(j \to f)_\pi$

$$L_j(\pi) < L_f(\pi) \tag{12}$$

is true, and for all $j \in N \setminus \{f, s\}$, satisfying the condition $(j \to s)_\pi$,

$$L_j(\pi) < L_s(\pi), \tag{13}$$

where $f = f(N,t)$ and $s = s(N,t)$, is also true.

Proof: For each job j such that $(j \to f)_\pi$, completion time $C_j(\pi) < C_f(\pi)$. If $d_j \ge d_f$, then obviously

$$L_j(\pi) = C_j(\pi) - d_j < C_f(\pi) - d_f = L_f(\pi), \tag{14}$$

therefore (12) is valid.

If for job $j \in N$, $(j \to f)_\pi$, then $d_j < d_f$. Then $r_j > r_f$. If $r_j \le r_f$, then $r_j(t) \le r_f(t)$ and $r_f(t) = r(t)$, as follows from (7) and (10). Then $r_j(t) = r_f(t) = r(t)$ and $d_j < d_f$,

but this contradicts the definition of job f (10). Therefore, $r_j > r_f$. Its obvious that $C_j(\pi) - p_j < C_f(\pi) - p_f$ and, since $r_j > r_f$,

$$C_j(\pi) - p_j - r_j < C_f(\pi) - p_f - r_f, \tag{15}$$

$$C_j(\pi) - C_f(\pi) < p_j + r_j - p_f - r_f. \tag{16}$$

Since $d_j < d_f$, then (from (9)) $d_j - r_j - p_j \geq d_f - r_f - p_f$ or $d_j - d_f \geq r_j + p_j - r_f - p_f$, so $C_j(\pi) - C_f(\pi) < p_j + r_j - p_f - r_f \leq d_j - d_f$. Then, $L_j(\pi,t) < L_f(\pi,t)$ for each job j, $(j \to f)_\pi$.

The inequality (13) can be proved in a similar way.

For each job j satisfying the condition $(j \to s)_\pi$, we have $C_j(\pi) < C_s(\pi)$. If $d_j \geq d_s$, then $L_j(\pi,t) = C_j(\pi) - d_j < C_s(\pi) - d_s = L_s(\pi,t)$, therefore (13) is true.

Let for the job $j \in N \setminus \{f\}$, $(j \to s)_\pi$, $d_j < d_s$, then $r_j > r_s$. Indeed, if we assume that $r_j \leq r_s$, then $r_j(t) \leq r_s(t)$ (it follows from (7)). In addition, $r_s(t) \geq r(t)$ for any job s according to definitions (8) and (11). If $r_s(t) = r(t)$, then for the jobs j and s we can write $r_j(t) = r_s(t) = r(t)$ and $d_j < d_s$, which contradicts the definition (11) of job $s(N,t)$. If $r_s(t) > r(t)$, that is, $r_s > r(t)$, then there is no job $i \in N \setminus \{f,s\}$, for which $r_s > r_i > r(t)$. Therefore, for the jobs j and s we get $r_j(t) = r_s(t)$ and $d_j < d_s$, which contradicts the definition (11) of job $s(N,t)$. Therefore, $r_j > r_s$.

Since $C_j(\pi) \leq C_s(\pi) - p_s$ and $p_j > 0$, then $C_j(\pi) - p_j < C_s(\pi) - p_s$ and since $r_j > r_s$, therefore $C_j(\pi) - p_j - r_j < C_s(\pi) - p_s - r_s$ and

$$C_j(\pi) - C_s(\pi) < p_j + r_j - p_s - r_s. \tag{17}$$

Since $d_j < d_s$, then from (9) we have $d_j - r_j - p_j \geq d_s - r_s - p_s$ or

$$C_j(\pi) - C_s(\pi) < p_j + r_j - p_s - r_s \leq d_j - d_s. \tag{18}$$

Hence, $L_j(\pi) < L_s(\pi)$ for each job $j \in N \setminus \{f\}$, $(j \to s)_\pi$.

Theorem 1.1 If conditions (9) are true for jobs in the subset $N' \subseteq N$, then at any time $t' \geq t$ and any early schedule $\pi \in \Pi(N')$ there exists $\pi' \in \Pi(N')$ such that

$$L_{\max}(\pi', t') \leq L_{\max}(\pi, t'), \quad \text{and} \quad C_{\max}(\pi', t') \leq C_{\max}(\pi, t') \tag{19}$$

and one of the jobs $f = f(N', t')$ or $s = s(N', t')$ is at the first position in schedule π'. If $d_f \leq d_s$, then job f is at the first position in schedule π'.

Proof: Let $\pi = (\pi_1, f, \pi_2, s, \pi_3)$, where π_1, π_2 and π_3 are partial schedules of π. Then, we construct a schedule $\pi' = (f, \pi_1, \pi_2, s, \pi_3)$. From the definitions (7), (8), (10) we get $r_f(t') \leq r_j(t'), j \in N'$, hence $C_{\max}((f, \pi_1), t') \leq C_{\max}((\pi_1, f), t')$ and

$$C_{\max}(\pi', t') \leq C_{\max}(\pi, t'), \quad \text{and} \tag{20}$$

$$L_j(\pi', t') \leq L_j(\pi, t'), \quad \forall j \in \{(\pi_2, s, \pi_3)\}. \tag{21}$$

From the lemma 1.1 we have

$$L_j(\pi', t') < L_s(\pi', t'), \quad \forall j \in \{\pi_1\} \cup \{\pi_2\}. \tag{22}$$

Obviously, the following inequality is true for job f

$$L_f(\pi', t') \le L_f(\pi, t'). \tag{23}$$

From (20)–(23) we get $C_{\max}(\pi', t') \le C_{\max}(\pi, t')$ and $L_{\max}(\pi', t') \le L_{\max}(\pi, t')$.

Let $\pi = (\pi_1, s, \pi_2, f, \pi_3)$, that is, job s is before job f. Construct a schedule $\pi' = (s, \pi_1, \pi_2, f, \pi_3)$. Further proof may be repeated as for job f. The first part of the theorem is proved.

Let us assume $d_f \le d_s$ and the schedule $\pi = (\pi_1, s, \pi_2, f, \pi_3)$. Then, we construct a schedule $\pi' = (f, \pi_{11}, \pi_{12}, \pi_3)$, where π_{11}, π_{12} are schedules of jobs of the sets $\{\, j \in N' : j \in \{(\pi_1, s, \pi_2)\}, d_j < d_f \,\}$ and $\{\, j \in N' : j \in \{(\pi_1, s, \pi_2)\}, d_j \ge d_f \,\}$. Jobs in π_{11} and π_{12} are ordered according to nondecreasing release times r_j. From $d_s \ge d_f$ we can conclude that $s \in \{\pi_{12}\}$.

For each job $j \in \{\pi_{11}\}$ we have $d_j < d_f$. Of (9) we get $d_j - r_j - p_j \ge d_f - r_f - p_f$, hence $r_j + p_j < r_f + p_f$, $\forall j \in \{\pi_{11}\}$, and $C_{\max}((f, \pi_{11}), t') = r_f(t') + p_f + \sum_{j \in \{\pi_{11}\}} p_j$. Since jobs in schedule $\{\pi_{12}\}$ are sorted by nondecreasing release times, then $C_{\max}((f, \pi_{11}, \pi_{12}), t') \le C_{\max}((\pi_1, s, \pi_2, f), t')$. As a result

$$C_{\max}(\pi', t') \le C_{\max}(\pi, t'), \quad \text{and} \tag{24}$$

$$L_j(\pi', t') \le L_j(\pi, t'), \quad \forall j \in \{\pi_3\}. \tag{25}$$

Job $j \in \{\pi_{12}\}$ satisfies $d_j \ge d_f$ and $C_j(\pi', t') \le C_f(\pi, t')$, which means

$$L_j(\pi', t') \le L_f(\pi, t'), \quad \forall j \in \{\pi_{12}\}. \tag{26}$$

Since $s \in \{\pi_{12}\}$, then

$$L_s(\pi', t') \le L_f(\pi, t'). \tag{27}$$

From the lemma 1.1

$$L_j(\pi', t') \le L_s(\pi', t'), \quad \forall j \in \{\pi_{11}\}. \tag{28}$$

Moreover, it is obvious that

$$L_f(\pi', t') \le L_f(\pi, t'). \tag{29}$$

From inequalities (24)–(29) it follows that $C_{\max}(\pi', t') \le C_{\max}(\pi, t')$ and $L_{\max}(\pi', t') \le L_{\max}(\pi, t')$, the theorem is proved.

We call a schedule $\pi' \in \Pi(N)$ **effective** if there is no schedule $\pi \in \Pi(N)$ such that $L_{\max}(\pi) \le L_{\max}(\pi')$ and $C_{\max}(\pi) \le C_{\max}(\pi')$, that is, at least one inequality would be strict.

Thus, when constraints (9) are satisfied for jobs from the set N, then there is an effective schedule π', in which either the job $f = f(N, t)$, or $s = s(N, t)$ is present. Moreover, if $d_f \le d_s$, then there is an effective schedule π' with a priority of job f.

We define the set of schedules $\Omega(N, t)$ as a subset of $\Pi(N)$ consisting of $n!$ schedules. Schedule $\pi = (i_1, i_2, ..., i_n)$ belongs to $\Omega(N, t)$ if we choose job $i_k, k = 1, 2, ..., n$ as $f_k = f(N_{k-1}, C_{i_{k-1}})$ or $s_k = s(N_{k-1}, C_{i_{k-1}})$, where $N_{k-1} = N \backslash \{i_1, i_2, ..., i_{k-1}\}$, $C_{i_{k-1}} = C_{i_{k-1}}(\pi)$ and $N_0 = N$, $C_{i_0} = t$. For $d_{f_k} \le d_{s_k}$ it is true that $i_k = f_k$, so if $d_{f_k} > d_{s_k}$, then $i_k = f_k$ or $i_k = s_k$. Its obvious that the set of schedules $\Omega(N, t)$ contains at most 2^n schedules.that is, $p_{2i} > y \ge p_{2i-1}$.

Example 1.1

$$\begin{cases} n = 2m, t \le r_1 < r_2 < \ldots < r_n, \\ r_{2i-1} < r_{2i} + p_{2i} < r_{2i-1} + p_{2i-1}, 1 \le i \le m, \\ r_{2i-1} + p_{2i-1} + p_{2i} < r_{2i+1} < r_{2i} + p_{2i} + p_{2i-1} < r_{2i+2}, 1 \le i \le m-1, \\ r_{2i-1} + p_{2i-1} + p_{2i} - d_{2i-1} > y, 1 \le i \le m-1, \\ r_{2i} + p_{2i} + p_{2i-1} - d_{2i} \le y. \end{cases}$$

The set $\Omega(N,t)$ contains 2^m schedules. The value of y is used below in the text. The optimal solution of the problem $1|r_j, d_j = r_j + p_j, L_{max} \le y|C_{\max}$ is $\pi^* = (2,1,4,3,\ldots,n,n-1)$.

Theorem 1.2 If for the jobs of the subset $N' \subseteq N$, $|N'| = n'$, is true (9), then at any time $t' \ge t$ and any schedule $\pi \in \Pi(N')$ exists a schedule $\pi' \in \Omega(N',t')$ such that

$$L_{\max}(\pi',t') \le L_{\max}(\pi,t') \quad \text{and} \quad C_{\max}(\pi',t') \le C_{\max}(\pi,t'). \tag{30}$$

Proof: Let $\pi = (j_1, j_2, \ldots, j_{n'})$ be an arbitrary schedule. We denote the first l jobs of the schedule π as π_l, $l = 0,1,2,\ldots,n'$, where π_0 is an empty schedule, and $\overline{\pi}_l = (j_{l+1}, \ldots, j_{n'})$, then $\pi = (\pi_l, \overline{\pi}_l)$. We introduce $N_l = N' \backslash \{\pi_l\}$ and $C_l = C_{\max}(\pi_l, t')$. Suppose for some l, $0 \le l < n'$, π_l is the largest initial partial the schedule of some schedule from $\Omega(N',t')$. If $j_1 \ne f(N',t')$ and $j_1 \ne s(N',t')$, then $\pi_l = \pi_0$, $l = 0$, then the largest partial schedule is empty. Let us say $f = f(N_l, C_l)$ and $s = s(N_l, C_l)$. If $d_f > d_s$, then $j_{l+1} \ne f$ and $j_{l+1} \ne s$, moreover when $d_f \le d_s$, then $j_{l+1} \ne f$, since π_{l+1} is not an initial schedule of some schedule from $\Omega(N',t')$.

According to the theorem 1.1 for the jobs of the set $\{\overline{\pi}_l\}, \overline{\pi}_l \in \Pi(N_l)$, there is a schedule $\overline{\pi}'_l$ starting at time C_l, for which $L_{\max}(\overline{\pi}'_l, C_l) \le L_{\max}(\overline{\pi}_l, C_l)$, $C_{\max}(\overline{\pi}'_l, C_l) \le C_{\max}(\overline{\pi}_l, C_l)$, and $[\overline{\pi}'_l]_1 = (f \text{ or } s)$, moreover, with $d_f \le d_s$, true $[\overline{\pi}'_l]_1 = f$, where $[\sigma]_k$ is the job in the k-th place in schedule σ. Hence, $L_{\max}((\pi_l, \overline{\pi}'_l), t') \le L_{\max}((\pi_l, \overline{\pi}_l), t')$ and $C_{\max}((\pi_l, \overline{\pi}'_l), t') \le C_{\max}((\pi_l, \overline{\pi}_l), t')$.

Let us denote $\pi' = (\pi_l, \overline{\pi}'_l)$. A feature of schedule π' is that the first $l+1$ jobs are the same as first $l+1$ jobs of some schedule from the set $\Omega(N',t')$, and $L_{\max}(\pi',t') \le L_{\max}(\pi,t')$, $C_{\max}(\pi',t') \le C_{\max}(\pi,t')$.

After no more than n' sequential conversions (since schedule length $n' \le n$) of the original randomly selected schedule π we come to schedule $\pi' \in \Omega(N',t')$, for which $L_{\max}(\pi',t') \le L_{\max}(\pi,t')$ and $C_{\max}(\pi',t') \le C_{\max}(\pi,t')$. The theorem is proved.

We form the following partial schedule $\omega(N,t) = (i_1, i_2, \ldots, i_l)$. For each job $i_k, k = 1,2,\ldots,l$, we have $i_k = f_k$ and $d_{f_k} \le d_{s_k}$, where $f_k = f(N_{k-1}, C_{k-1})$ and $s_k = s(N_{k-1}, C_{k-1})$. For $f = f(N_l, C_l)$ and $s = s(N_l, C_l)$ inequality $d_f > d_s$ is true. In case when $d_f > d_s$ for $f = f(N,t)$ and $s = s(N,t)$, then $\omega(N,t) = \emptyset$. So $\omega(N,t)$ is the "maximum" schedule, during the construction of which job (like f) for the next place of the schedule can be uniquely selected. We can construct a schedule $\omega(N,t)$ for set of jobs N starting at time t using the algorithm 1.1.

Algorithm 1.1 for constructing schedule $\omega(N,t)$.

1: **Initial step.** Let $\omega = \emptyset$.
2: **Main step.** Find the jobs $f := f(N,t)$ and $s := s(N,t)$;
3: **if** $d_f \le d_s$ **then**
4: $\omega := (\omega, f)$
5: **else**

6: algorithm stops;
7: **end if**
8: Let $N := N \setminus \{f\}$, $t := r_f(t) + p_f$ and go to the next main step.

Lemma 1.2 The complexity of the algorithm 1.1 for finding the schedule $\omega(N,t)$ is at most $O(n \log n)$ operations for any N and any t.

Proof: At each iteration of the algorithm 1.1 we find two jobs: $f = f(N,t)$ and $s = s(N,t)$. If jobs are ordered by release times r_j (and, accordingly, time $r(t)$ is for $O(1)$ operations), then to find two jobs (f and s) you need $O(\log n)$ operations. Total number of iterations is not more than n. Thus, constructing a schedule $\omega(N,t)$ requires $O(n \log n)$ operations.

The main step of algorithm 1.1 is finding the jobs f and s and it requires at least $O(\log n)$ operations. Obviously, the number of iterations of the algorithm is $O(n)$, therefore, the complexity of the algorithm 1.1 of $O(n \log n)$ operations is the minimum possible for constructing the schedule $\omega(N,t)$.

Lemma 1.3 If for jobs of the set N conditions (9) are true, then any schedule $\pi \in \Omega(N,t)$ starts with the schedule $\omega(N,t)$.

Proof: If $\omega(N,t) = \emptyset$, that is, $d_f > d_s$, where $f = f(N,t)$, $s = s(N,t)$, the statement of the lemma is true, since any schedule starts from empty.

Let $\omega(N,t) = (i_1, i_2, ..., i_l)$, $l > 0$, and so for each i_k, $k = 1, 2, ..., l$, we have $i_k = f_k$ and $d_{f_k} \le d_{s_k}$, where $f_k = f(N_{k-1}, C_{k-1})$ and $s_k = s(N_{k-1}, C_{k-1})$. For $f = f(N_l, C_l)$ and $s = s(N_l, C_l)$ it is true that $d_f > d_s$. As seen from the definition of the set of schedules $\Omega(N,t)$ all schedules in this subset start with a partial schedule $\omega(N,t)$.

Let us use the following notation $\omega^1(N,t) = (f, \omega(N', t'))$ and $\omega^2(N,t) = (s, \omega(N'', t''))$, where $f = f(N,t)$, $s = s(N,t)$, $N' = N \setminus \{f\}$, $N'' = N \setminus \{s\}$, $t' = r_f(t) + p_f$, $t'' = r_s(t) + p_s$. Obviously, the algorithm for finding ω^1 (as well as ω^2) requires $O(n \log n)$ operations, as much as the algorithm for constructing $\omega(N,t)$.

Consequence 1.1 **from Lemma 1.3.** If the jobs of the set N satisfy conditions (9), then each schedule $\pi \in \Omega(N,t)$ starts either with $\omega^1(N,t)$, or with $\omega^2(N,t)$.

Theorem 1.3 If the jobs of the set N satisfy conditions (9), then for any schedule $\pi \in \Omega(N,t)$ it is true that $(i \to j)_\pi$ for any $i \in \{\omega^1(N,t)\}$ and $j \in N \setminus \{\omega^1(N,t)\}$.

Proof: In the case $\{\omega^1(N,t)\} = N$ statement of the theorem is obviously true. Let $\{\omega^1(N,t)\} \ne N$. Further in in the proof we use the notation $\omega^1 = \omega^1(N,t)$.

If $f = f(N,t)$ and $s = s(N,t)$ are such that $d_f \le d_s$, then all schedules from the set $\Omega(N,t)$ begin with a partial schedule $\omega(N,t) = \omega^1$, therefore the statement of the theorem is also true.

Consider the case of $d_f > d_s$. All schedules from set $\Omega(N,t)$ starting with job f have partial schedule $\omega(N,t) = \omega^1$.

Let us choose any arbitrary schedules $\pi \in \Omega(N,t)$ with job s comes first, $\pi_1 = s$, and any schedule $|\omega^1| = l$, $l < n$, containing l jobs. Let $\pi_l = (j_1, j_2, ..., j_l)$ be a partial schedule of schedule π containing l jobs, where $j_1 = s$. We need to prove that $\{\pi_l\} = \{\omega^1\}$. Let us assume the contrary that there is a job $j \in \{\pi_l\}$, but $j \notin \{\omega^1\}$.

For case $(j \to f)_\pi$ we need to check two subcases. If $d_j < d_f$, then from (9) we have $d_j - r_j - p_j \ge d_f - r_f - p_f$, therefore $r_j + p_j < r_f + p_f$. Then job j is included in schedule ω^1 according to the definition of $\omega(N,t)$ and ω^1, but by our assumption $j \notin \{\omega^1\}$. If $d_j \ge d_f$, then from the fact that $\pi \in \Omega(N,t)$ follows $(f \to j)_\pi$, but this contradicts $(j \to f)_\pi$. Therefore, $j \in \{\omega^1\}$.

The other case is $(f \to j)_\pi$. Then for each job $i \in \{\omega^1\}$, for which $i \notin \{\pi_l\}$, conditions $r_i < r_i + p_i \le C_{\max}(\omega^1) < r_{s_{l+1}} \le r_j$ are true, because $j \notin \{\omega^1\}$,

where $s_{l+1} = s\left(N\backslash\{\omega^1\}, C_{\max}(\omega^1)\right)$. Jobs s_{l+1} and j were not ordered in schedule ω^1, therefore, $C_{\max}(\omega^1) < r_{s_{l+1}} \le r_j$. Besides, $d_i \le d_j$. If $d_i > d_j$, then $r_i + p_i \ge r_j + p_j$, but $r_i + p_i < r_j$ is true. Hence $(i \to j)_{\pi_l}$, since $\pi = (\pi_l, \overline{\pi}_l) \in \Omega(N,t)$, but it contradicts our guess that $i \notin \{\pi_l\}$ and $j \in \{\pi_l\}$.

Therefore, our assumption is not true and $\{\omega^1\} = \{\pi_l\}$. The theorem is proved.

Therefore, jobs of the set $\{\omega^1(N,t)\}$ precede jobs of the set $N\backslash\{\omega^1(N,t)\}$ for any schedule from the set $\Omega(N,t)$, including the optimal schedule.

3.2 Performance problem with constraint on maximum lateness

The problem $1|d_i \le d_j, d_i - r_i - p_i \ge d_j - r_j - p_j; L_{\max} \le y|C_{\max}$ consists of the following. We need to find schedule θ for any y with $C_{\max}(\theta) = \min\{C_{\max}(\pi) : L_{\max}(\pi) \le y\}$. If $L_{\max}(\pi) > y$ for any $\pi \in \Pi(N)$, then $\theta = \emptyset$.

Lemma 1.4 The complexity of algorithm 1.2 does not exceed $O(n^2 \log n)$ operations.

Proof: At each iteration of the main step of the algorithm 1.2 we find the schedules ω^1 and, if necessary, ω^2 in $O(n \log n)$ operations. Since ω^1 and ω^2 consist of at least one job, then at each iteration of the algorithm we either add one or mere jobs to the schedule θ, or assume $\theta = \emptyset$ and stop. Therefore, the total number of steps of the algorithm is at most n. Thus, algorithm 1.2 requires $O(n^2 \log n)$ operations.

Algorithm 1.2 for solving the problem $1|d_i \le d_j, d_i - r_i - p_i \ge d_j - r_j - p_j; L_{\max} \le y|C_{\max}$.

1: **Initial step.** Let $\theta := \omega(N,t), t' := t$;
2: **Main step.**
3: **if** $L_{\max}(\theta, t') > y$ **then**
4: $\theta := \emptyset$ and the algorithm stops.
5: **end if**
6: Find $N' := N\backslash\{\theta\}$, $t' := C_{\max}(\theta)$ and $\omega^1(N', t'), \omega^2(N', t')$.
7: **if** $N' = \emptyset$ **then**
8: the algorithm stops.
9: **else**
10: **if** $L_{\max}(\omega^1, t') \le y$ **then**
11: $\theta := (\theta, \omega^1)$ and go to next step;
12: **end if**
13: **if** $L_{\max}(\omega^1, t') > y$ and $L_{\max}(\omega^2, t') \le y$ **then**
14: $\theta := (\theta, \omega^2)$ and go to next step;
15: **end if**
16: **if** $L_{\max}(\omega^1, t') > y$ and $L_{\max}(\omega^2, t') > y$ **then**
17: $\theta := \emptyset$ and the algorithm stops.
18: **end if**
19: **end if**

The problem $1|d_i \le d_j, d_i - r_i - p_i \ge d_j - r_j - p_j; L_{\max} \le y|C_{\max}$ cannot be solved in less than $O(n^2 \log n)$ operations because there exists (*Example 1.1*). The optimal schedule for this example is $\pi^* = (2, 1, 4, 3, \ldots, n, n-1)$. To find this schedule, we need $O(n^2 \log n)$ operations.

We denote by $\theta(N,t,y)$ the schedule constructed by algorithm 1.2 starting at time t from the jobs of the set N with the maximum lateness not more than y. If $N=\emptyset$, then $\theta(\emptyset,\mathrm{t},\mathrm{y})=\emptyset$ for any t and y.

Theorem 1.4 Let the jobs of the set N satisfy conditions (9). If the algorithm 1.2 constructs the schedule $\theta(N,t,y)\neq\emptyset$, then $C_{\max}(\theta)=$ $\min\{C_{\max}(\pi):L_{\max}(\pi)\leq y,\pi\in\Pi(N)\}$. If, as a result of the algorithm 1.2 the schedule will not be generated, that is, $\theta(N,t,y)=\emptyset$, then $L_{\max}(\pi)>y$ for each $\pi\in\Pi(N)$.

Proof: In case if for schedule $\pi\in\Pi(N)$ condition $L_{\max}(\pi)\leq y$ is true, then according to Theorem 1.2 there is a schedule $\pi'\in\Omega(N,t)$ such that $L_{\max}(\pi')\leq L_{\max}(\pi)\leq y$ and $C_{\max}(\pi')\leq C_{\max}(\pi)$. Therefore, the required schedule θ contains in set $\Omega(N,t)$.

According to Lemma 1.3, all schedules of the set $\Omega(N,t)$ start with $\omega(N,t)$. Let us take $\theta_0=\omega(N,t)$.

After $k, k\geq 0$ main steps of the algorithm 1.2 we got the schedule θ_k and $N'=N\backslash\{\theta_k\}$, $t'=C_{\max}(\theta_k)$. Let us assume that there is an optimal by the criterion of maximum completion time ($C_{\max}$) schedule θ starting with θ_k. According to Theorem 1.2, there is an optimal extension of the schedule θ_k among the schedules from the set $\Omega(N',t')$.

Let $\theta_{k+1}=(\theta_k,\omega^1(N',t'))$, that is, $L_{\max}(\theta_{k+1})\leq y$. According to Theorem 1.3, for schedule ω^1, $\omega^1=\omega^1(N',t')$, there is no artificial idle times of the machine and all schedules from the set $\Omega(N',t')$ start with jobs of the set $\{\omega^1(N',t')\}$. Therefore, $\omega^1(N',t')$ is the best by the criterion of $C_{\max}$ among all feasible by maximum lateness ($L_{\max}$) extensions of the partial schedule θ_k.

If $\theta_{k+1}=(\theta_k,\omega^2(N',t'))$, that is, $L_{\max}(\omega^1,t')>y$, and $L_{\max}(\omega^2,t')\leq y$. All schedules of the set $\Omega(N',t')$ start with either schedule $\omega^1(N',t')$ or $\omega^2(N',t')$. As $L_{\max}(\omega^1,t')>y$, then the only suitable extension is $\omega^2(N',t')$.

Thus, at each main step of the algorithm, we choose the fastest continuation of the partial schedule θ_k among all those allowed by the maximum lateness. After no more than n main steps of the algorithm, the required schedule is constructed.

Let us assume that after the $k+1$ steps of the algorithm $L_{\max}(\omega^1,t')>y$ and $L_{\max}(\omega^2,t')>y$. If schedule θ could exist, that is, $\theta\neq\emptyset$, then θ would start with θ_k. Then for any schedule $\pi\in\Pi(N',t')$ there would exist a schedule $\pi'\in\Omega(N',t')$ such that $L_{\max}(\pi,t')\geq L_{\max}(\pi',t')\geq L_{\max}(\omega^1,t')>y$ or $L_{\max}(\pi,t')\geq L_{\max}(\pi',t')\geq L_{\max}(\omega^2,t')>y$. Therefore $\theta=\emptyset$.

Repeating our proof as many times as the main step of algorithm 1.2 (no more than n), we come to the truth of the statement of the theorem.

3.3 Algorithm for constructing a set of Pareto schedules by criteria $C_{\max}$ and $L_{\max}$

Let us develop an algorithm for constructing a set of Pareto schedules $\Phi(N,t)=\{\pi'_1,\pi'_2,\ldots,\pi'_m\}$, $m\leq n$, by criteria $C_{\max}$ and $L_{\max}$ according to conditions (5)–(6).

Schedule π'_m is a solution to problem $1|r_j|L_{\max}$ if (9) is true.

Algorithm 1.3 for constructing a set of Pareto schedules by criteria $C_{\max}$ and $L_{\max}$.

1: **Initial step.** $Y:=+\infty$, $\pi^*:=\omega(N,t)$, $\Phi:=\emptyset$, $m:=0$, $N':=N\backslash\{\pi^*\}$ and $t':=C_{\max}(\pi^*)$.

2: **if** $N'=\emptyset$ **then**

3: $\Phi:=\Phi\cup(\pi^*)$, $m:=1$ and the algorithm stops.

4: **end if**

5: **Main step.**

6: **if** $L_{\max}(\omega^1, t') \le L_{\max}(\pi^*)$ **then**
7: $\pi^* := (\pi^*, \omega^1)$, where $\omega^1 = \omega^1(N', t')$ and go to the next step;
8: **end if**
9: **if** $L_{\max}(\omega^1, t') > L_{\max}(\pi^*)$ **then**
10: **if** $L_{\max}(\omega^1, t') < y$ **then**
11: find $\theta = \theta(N', t', y')$ using algorithm 1.2, where $y' = L_{\max}(\omega^1, t')$;
12: **if** $\theta = \emptyset$ **then**
13: $\pi^* := (\pi^*, \omega^1)$ and go to the next step;
14: **else**
15: $\pi' := (\pi^*, \theta)$
16: **if** $C_{\max}(\pi'_m) < C_{\max}(\pi')$ **then**
17: $m := m + 1$, $\pi'_m := \pi'$, $\Phi := \Phi \cup (\pi'_m)$, $y = L_{\max}(\pi'_m)$;
18: **else**
19: $\pi'_m = \pi'$ and go to next step;
20: **end if**
21: **end if**
22: **if** $L_{\max}(\omega^1, t') \ge y$ **then**
23: find $\omega^2 = \omega^2(N', t')$;
24: **if** $L_{\max}(\omega^2, t') < y$ **then**
25: $\pi^* = (\pi^*, \omega^2)$ and go to the next step;
26: **else**
27: $\pi^* = \pi_m{}'$ and the algorithm stops.
28: **end if**
29: **end if**
30: **end if**
31: **end if**

As a result of the algorithm 1.3, a set of schedules $\Phi(N, t)$ is constructed, for the set of jobs N starting at time t, for which inequality $1 \le |\Phi(N, t)| \le n$ true. We should note that the set $\Phi(N, t)$ for *Example 1.1* consists of two schedules, although set $\Omega(N, t)$ consists of $2^{\frac{n}{2}}$ schedules:

$$\pi_{1'} = (1, 2, 3, 4, ..., n-1, n), \tag{31}$$

$$\pi_{2'} = (2, 1, 4, 3, ..., n, n-1). \tag{32}$$

Lemma 1.5 The complexity of the algorithm 1.3 does not exceed $O(n^3 \log n)$ operations.

Proof: At each iteration of the main step of the algorithm 1.3 we find schedules ω^1 and, if necessary, ω^2, which requires $O(n \log n)$ operations according to lemma 1.2, and also schedule θ in $O(n^2 \log n)$ operations. As ω^1 and ω^2 consist of at least one job, then at any iteration of the algorithm one or more jobs are added to the schedule π^*, or the algorithm stops at last schedule π'. Therefore, the total number of iterations is at most n. Thus, it takes no more than $O(n^3 \log n)$ operations to execute algorithm 1.3.

Theorem 1.5 If case if (9) is true for each job of the set N, then the schedule π^*, constructed by algorithm 1.3, is optimal according to the criterion $L_{\max}$. Moreover, for any schedule $\pi \in \Pi(N)$ there exists a schedule $\pi' \in \Phi(N, t)$ such that $L_{\max}(\pi') \le L_{\max}(\pi)$ and $C_{\max}(\pi') \le C_{\max}(\pi)$.

Proof: According to Theorem 1.2, there exists an optimal (by $L_{\max}$) schedule from set $\Omega(N, t)$. According to Lemma 1.3, all schedules of the set $\Omega(N, t)$ start with a partial schedule $\omega(N, t)$.

Let $\pi_0 = \omega(N,t)$. After k, $k \geq 0$, main steps of algorithm 1.3 we have a partial schedule π_k. Suppose there is an optimal (by $L_{\max}$) schedule starting with π_k. We denote $N' = N \backslash \{\pi_k\}$ and $t' = C_{\max}(\pi_k)$.

If $\pi_{k+1} = (\pi_k, \omega^1)$, where $\omega^1 = \omega^1(N't')$, then either $L_{\max}(\omega^1, t') \leq L_{\max}(\pi_k)$, or $L_{\max}(\pi_k) < L_{\max}(\omega^1, t') < y$, that is, current value of the criterion and the maximum lateness will "appear" on next steps of the algorithm 1.3. That is, $\theta(N', t', y') = \emptyset$, where $y' = L_{\max}(\omega^1, t')$. If $\theta = \theta(N', t', y') \neq \emptyset$, then we improve the current maximum lateness value: $\pi' = (\pi_k, \theta)$ and $y = L_{\max}(\pi') = L_{\max}(\omega^1, t')$. The schedule π' is added to the set of schedules $\Phi(N,t)$. Moreover, according to Theorem 1.3 jobs of set $\{\omega^1\}$ precede jobs of set $N' \backslash \{\omega^1\}$. Thus, the schedule ω^1alert(without artificial idle times of the machine) would be the best continuation for π_k.

If $\pi_{k+1} = (\pi_k, \omega^2)$, where $\omega^2 = \omega^2(N', t')$, that is, according to algorithm 1.3 $L_{\max}(\omega^2, t') < L_{\max}(\pi') \leq L_{\max}(\omega^1, t')$. In this case the continuation ω^2 is "better" than ω^1. Hence, the partial schedule π_{k+1} is a part of some optimal schedule.

Repeating our proof no more than n times, we come to optimality (for $L_{\max}$) of the schedule π^*.

The set of schedules $\Phi(N,t)$ contains at most n schedules, since at each main step of the algorithm in the set $\Phi(N,t)$ at most one schedule is "added," and this step is executed no more than n times.

Suppose there is a schedule $\pi \in \Pi(N), \pi \notin \Phi(N,t),$ such that either $C_{\max}(\pi) \leq C_{\max}(\pi')$ and $L_{\max}(\pi) \geq L_{\max}(\pi')$, or $C_{\max}(\pi) \geq C_{\max}(\pi')$ and $L_{\max}(\pi) \leq L_{\max}(\pi')$ for each schedule $\pi' \in \Phi(N,t)$. Moreover, in each pair of inequalities at least one inequality is strict. According to Theorem 1.1, there is a schedule $\pi'' \in \Omega(N,t)$ such that $L_{\max}(\pi'') \leq L_{\max}(\pi)$ and $C_{\max}(\pi'') \leq C_{\max}(\pi)$. If $\pi'' \in \Phi(N,t)$. Thus, it becomes obvious that our assumption is not correct. Let $\pi'' \in \Omega(N,t) \backslash \Phi(N,t)$. Algorithm 1.3 shows that the structure of each schedule $\pi' \in \Phi(N,t)$ can be represented as a sequence of partial schedules $\pi' = \left(\omega'_0, \omega'_1, \omega'_2, ..., \omega'_{k'}\right),$ where $\omega'_0 = \omega(N,t)$, and ω'_i is either $\omega^1(N'_i, C'_i),$ or $\omega^2(N'_i, C'_i),$ and $N'_i = N \backslash \{\omega'_0, ..., \omega'_{i-1}\}$, $C'_i = C_{\max}\left((\omega'_0, ..., \omega'_{i-1}), t\right), i = 1, 2, ..., k'$. The schedule π'' has the same structure according to the definition of the set $\Omega(N,t)$, that is, $\pi = (\omega''_0, \omega''_1, \omega''_2, ..., \omega''_{k''}),$ possibly $k'' \neq k',$ where $\omega''_0 = \omega'_0 = \omega(N,t), \omega''_i$ is equal to either $\omega^1(N''_i, C''_i),$ or $\omega^2(N''_i, C''_i),$ a $N''_i = N \backslash \{\omega''_0, ..., \omega''_{i-1}\}$, $C''_i = C_{\max}\left((\omega''_0, ..., \omega''_{i-1}), t\right),$ $i = 1, 2, ..., k''$.

We assume that the first k partial schedules π'' and π' are equal, that is, $\omega''_i = \omega'_i = \omega_i, i = 0, 1, ..., k-1, \omega''_k \neq \omega'_k$. If $y = L_{\max}(\omega_0, ..., \omega_{k-1}),$ let us construct a schedule θ using algorithm 1.2, $\theta = \theta(N_k, C_k, y)$. If $\theta = \emptyset,$ then according to algorithm 1.3, $\omega'_k = \omega^1(N_k, C_k)$. Because of $\omega''_k \neq \omega'_k,$ schedule $\omega''_k = \omega^2(N_k, C_k)$. Objective function value ($L_{\max}$) can be reached on a job from the set $N_k,$ since $\theta = \emptyset$. The whole structure of the algorithm 1.3 construct in such a way that up to the "critical" job (according to $L_{\max}$) order the jobs as "tightly" as possible, therefore we complete the schedule ω^1, after which $C_{\max}(\pi') \leq C_{\max}(\pi'')$ and $L_{\max}(\pi') \leq L_{\max}(\pi'')$. If $\theta \neq \emptyset$, then for schedules π' and π'' $C_{\max}(\pi') \leq C_{\max}(\pi'')$ and $L_{\max}(\pi') = L_{\max}(\pi'')$. Thus, for any schedule $\pi'' \in \Omega(N,t) \backslash \Phi(N,t)$ exists schedule $\pi' \in \Phi(N,t)$ such that $C_{\max}(\pi') \leq C_{\max}(\pi'')$ and $L_{\max}(\pi') \leq L_{\max}(\pi'')$. Hence, for any schedule $\pi \in \Pi(N)$ there exists schedule $\pi' \in \Phi(N,t)$ such that $L_{\max}(\pi') \leq L_{\max}(\pi)$ and $C_{\max}(\pi') \leq C_{\max}(\pi)$. The theorem is proved.

Figure 1 schematically shows the considered schedule.

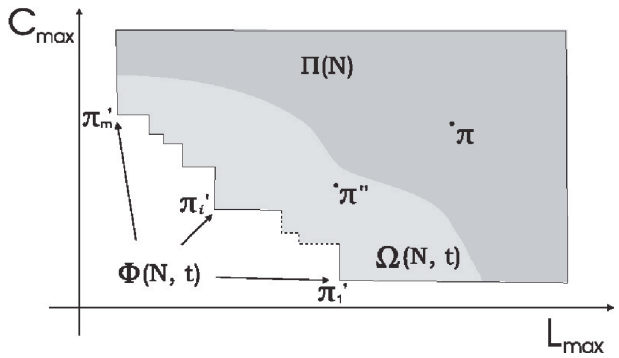

Figure 1.
The set of Pareto-optimal schedules.

For the set of schedules $\Phi(N,t) = \{\pi'_1, \pi'_2, ..., \pi'_m\}$, $m \le n$, we conditions (5)–(6) are true.

The schedule π'_1 is optimal in terms of speed ($C_{\max}$), and π'_m is optimal in terms of the maximum lateness (by $L_{\max}$) if the jobs of the set N satisfy the conditions (9).

4. Conclusions

Single machine scheduling problem with given release dates and two objective functions is considered in this chapter, which is *NP*-hard in the strong sense.
A number of new polynomially and pseudo-polynomially solvable subcases of the problem were found. For a case when

$$d_1 \le ... \le d_n, \quad d_1 - r_1 - p_1 \ge ... \ge d_n - r_n - p_n, \tag{33}$$

an algorithm for constructing a Pareto-optimal set of schedules by criteria $C_{\max}$ and $L_{\max}$ is developed. It is proved that the complexity of the algorithm does not exceed $O(n^3 \log n)$ operations.

An experimental study of the algorithm showed that it can be used to construct optimal schedules (by $L_{\max}$) even for instances not satisfying the conditions (33).

Acknowledgements

The research was supported by RFBR (project 20-58-S52006).

Author details

Alexander A. Lazarev and Nikolay Pravdivets*
Institute of Control Sciences, Moscow, Russia

*Address all correspondence to: pravdivets@ipu.ru

References

[1] Lenstra JK, Rinnooy Kan AHG, Brucker P. Complexity of machine scheduling problems. Annals of Discrete Mathematics. 1977;**1**:343-362

[2] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics. 1979;**5**: 287-326

[3] Potts CN. Analysis of a heuristic for one machine sequencing with release dates and delivery times. Operations Research. 1980;**28**:1436-1441

[4] Jackson JR. Scheduling a Production Line to Minimize Maximum Tardiness. Los Angeles, CA: University of California; 1955. Manag. Sci. Res. Project. Research Report N 43

[5] Hall LA, Shmoys DB. Jackson's rule for one-machine schedulings: Making a good heuristic better. Mathematics of Operations Research. 1992;**17**:22-35

[6] Mastrolilli M. Efficient approximation schemes for scheduling problems with release dates and delivery times. Journal of Scheduling. 2003;**6**(6):521-531

[7] Garey MR, Johnson DS, Simons BB, Tarjan RE. Scheduling UnitTime tasks with arbitrary release times and deadlines. SIAM Journal on Computing. 1981;**10**:256-269

[8] Vakhania N. Single-machine scheduling with release times and tails. Annals of Operations Research. 2004; **129**(1–4):253-271

[9] Vakhania N. Dynamic restructuring framework for Scheduling with release times and due-dates. Mathematics. 2019;**7**(11):1104

[10] Lawler EL. Optimal sequencing of a single machine subject to precedence constraints. Management Science. 1973; **19**(5):544-546

[11] Simons BB. A fast algorithm for single processor scheduling. In: Proceedings of the 19th IEEE Annual Symposium on Foundations of Computer Science. New York: Ann. Arbor. Mich; 1978. pp. 246-252

[12] Baker KR, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Preemtive scheduling of a single machine to minimize maximum cost subject to release dates and precedence constraints. Operations Research. 1983; **31**(2):381-386

[13] Hoogeveen JA. Minimizing maximum promptness and maximum lateness on a single machine. Mathematics of Operations Research. 1996;**21**:100-114

[14] Lazarev AA, Shulgina ON. Polynomially solvable subcases of the problem of minimizing maximum lateness. Izvestiya VUZov. Mathematics. 2000 (in Russian)

[15] Vakhania N. Scheduling a single machine with primary and secondary objectives. Algorithms. 2018;**11**(6):80

[16] Vakhania N. Fast solution of single-machine scheduling problem with embedded jobs. Theoretical Computer Science. 2019;**782**:91-106