mathematics

MDPI

Article

On the Dual and Inverse Problems of Scheduling Jobs to Minimize the Maximum Penalty

Alexander A. Lazarev [1], Nikolay Pravdivets [1,*] and Frank Werner [2]

[1] Institute of Control Sciences, 117997 Moscow, Russia; jobmath@mail.ru
[2] Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, 39106 Magdeburg, Germany; frank.werner@ovgu.de
* Correspondence: pravdivets@ipu.ru; Tel.: +7-495-334-87-51

Received:11 June 2020; Accepted: 6 July 2020; Published: 10 July 2020

Abstract: In this paper, we consider the single-machine scheduling problem with given release dates and the objective to minimize the maximum penalty which is NP-hard in the strong sense. For this problem, we introduce a dual and an inverse problem and show that both these problems can be solved in polynomial time. Since the dual problem gives a lower bound on the optimal objective function value of the original problem, we use the optimal function value of a sub-problem of the dual problem in a branch and bound algorithm for the original single-machine scheduling problem. We present some initial computational results for instances with up to 20 jobs.

Keywords: single-machine scheduling; minimization of maximum penalty; dual problem; inverse problem; branch and bound

MSC: 90B35; 90C57

1. Introduction

We consider single-machine scheduling problems, where a set of n jobs $N = \{1, 2, \ldots, n\}$ has to be processed on a single machine starting at time τ. For each job j, a release date r_j, a processing time p_j and a due date d_j are given. Many scheduling problems require the minimization of some maximum term. Denote by C_j the completion time of job j, then the minimization of the makespan (i.e., the maximum completion time of the jobs)

$$C_{max} = \max_{j=1,\ldots,n} C_j,$$

or the minimization of maximum lateness

$$L_{max} = \max_{j=1,\ldots,n} \{C_j - d_j\}$$

are well-known examples of such an optimization criterion.

In this paper, we consider two related problems of such a min-max problem, namely a dual problem as well as an inverse problem of the single-machine scheduling problem with given release dates and minimizing the maximum penalty. While the original problem $1|r_j|L_{max}$ is NP-hard in the strong sense [1], we prove that both the dual and inverse problems of this problem can be solved in polynomial time.

Due to the NP-hardness of the problem $1|r_j|L_{max}$, several branch and bound algorithms have been developed and special cases of the problem have been considered, see e.g., [2–9]. In [9], it has been shown that if the release dates for all jobs are from the interval $[d_j - p_j - A, d_j - A]$ for all jobs and

some constant A, the problem can be solved in $O(n \log n)$ time if no machine idle times are allowed and in $O(n^2 \log n)$ time if machine idle times are allowed. Another important special case of this problem has been considered in [10]. In that paper, it is shown that for naturally bounded job data, the problem can be polynomially solved. More precisely, a polynomial time solution of the variant is given when the maximal job processing time and the differences between the job release dates are bounded by a constant. The binary search procedure presented in this work determines an optimal solution in $O(n^2 \log n \log p_{max})$ or $O(d_{max} n \log n \log p_{max})$ time, where p_{max} is the maximal processing time and d_{max} is the maximal due date.

In [11], some computational experiences and applications for the more general case with additional precedence constraints are reported. Their algorithm turned out to be superior to earlier algorithms. Some applications to job shop scheduling are also discussed. A more recent branch and bound algorithm for this single-machine problem with release dates and precedence constraints has been given in [12]. This algorithm uses four heuristics for finding initial upper bounds and a variable neighborhood search procedure. It solved most considered instances within one minute of CPU time. Several approximation schemes for four variants of the problem with additional non-availability or deadline constraints have been derived in [13]. An approximation algorithm for this single-machine problem with additional workload dependent maintenance duration has been presented in [14]. This algorithm is even optimal for some special cases of the problem. A hybrid metaheuristic search algorithm for the single-machine problem with given release dates and precedence constraints has been developed in [15]. Computational tests have been made using own instance sets with 100 jobs and instances from [12] with up to 1000 jobs. The hybridization of the elektromagnetism algorithm with tabu search leads to a tradeoff between diversification and intensification strategies. The metric approach is another recent possibility for solving the problem $1|r_j|L_{max}$ approximately with guaranteed maximal error, see e.g., [16]. The introduced metric delivers an upper bound on the absolute error of the objective function value. Taking a given instance of some problem and using the introduced metric, the nearest instance is determined for which a polynomial or pseudo-polynomial algorithm is known. Then a schedule is constructed for this instance and applied to the original instance.

There have been also considered problems with several optimization criteria. In [17], the single-machine problem with the primary criterion of minimizing maximum lateness and the secondary criterion of minimizing the maximum job completion time has been investigated. The author gives dominance properties and conditions when the Pareto-optimal set can be found in polynomial time. The derived properties allow extension of the basic framework to exponential implicit enumeration schemes and polynomial approximation algorithms. The problem of finding the Pareto-optimal set for two criteria in the case when there are constraints on the source data have been considered in [18,19]. In [18], the idea of the dual approach is considered in detail, but there is no sufficient experimental study of the effectiveness of this approach. Lazarev et al. [20] considered the problem of minimizing maximum lateness and the makespan in the case of equal processing times and proposed a polynomial time approach for finding the Pareto-optimal set of feasible solutions. They presented two approaches, the efficiency of which depends on the number of jobs and the accuracy of the input-output parameters.

The dual and inverse problems considered in this paper are maximization problems. In the literature, there exist some works on other single-machine maximization problems, usually under the assumption of no inserted idle times between the processing of jobs on the machine. Maximization problems in single-machine problems were considered e.g., in [21,22]. The complexity and some algorithms for single-machine total tardiness maximization problems have been discussed in [23]. In [24], a pseudo-polynomial algorithm for the single-machine total tardiness maximization problem has been transformed by a graphical algorithm into a polynomial one.

The remainder of this paper is as follows. In Section 2, we introduce the dual problem of the single-machine problem, where the maximum penalty term of a job should be minimized. Section 3 considers an inverse problem, where the minimum of the penalty terms should be maximized. The solution of this dual problem is embedded into a branch and bound algorithm for the original

problem. Some computational results for this branch and bound algorithm are given in Section 4. The paper finishes with some concluding remarks.

2. The Dual Problem

Let us consider the general formulation of the NP-hard problem of minimizing the maximum penalty or cost $\varphi_{\max}$ for a set of jobs on a single machine, i.e., problem $1 \mid r_j \mid \varphi_{\max}$. The machine cannot process more than one job of the set $N = \{1, \ldots, n\}$ at a moment. Preemptions of the processing of a job are prohibited. Let $\varphi_{j_k}(C_{j_k}(\pi))$ denote the penalty for job $j \in N$ if it is processed as the k-th job in the sequence $\pi = (j_1, j_2, \ldots, j_k, \ldots, j_n)$. We assume that all $\varphi_{j_k}(C_{j_k}(\pi)), k = 1, 2, \ldots, n$, are arbitrary non-decreasing penalty functions.

By μ^* we denote the optimal value of the objective function:

$$\mu^* = \min_{\pi \in \Pi(N)} \max_{k=1,\ldots,n} \varphi_{j_k}(C_{j_k}(\pi)), \tag{1}$$

where $\Pi(N) = \{\pi_1, \pi_2, \ldots, \pi_{n!}\}$ denotes the set of all permutations (schedules) of the jobs of the set N.

In the scheduling literature, many special cases of the following general dual problem are considered. One wishes to find an optimal job sequence π^* and the corresponding objective function value ν^* such that

$$\nu^* = \max_{k=1,\ldots,n} \min_{\pi \in \Pi(N)} \varphi_{j_k}(C_{j_k}(\pi)). \tag{2}$$

For convenience, we introduce a notation that takes into account the position of the job in the schedule. Let a schedule (job sequence) $\pi \in \Pi(N)$ be given by $\pi = (j_1, j_2, \ldots, j_n)$. For the job, which is processed as the k-th job in the sequence, $k = 1, 2, \ldots, n$, under a schedule π, we denote:

$$\nu_k = \min_{\pi \in \Pi(N)} \varphi_{j_k}(C_{j_k}(\pi)), \; k = 1, 2, \ldots, n. \tag{3}$$

Obviously,

$$\nu^* = \max_{k=1,\ldots,n} \nu_k. \tag{4}$$

Lemma 1 ([25]). *Let $\varphi_j(t), j = 1, 2, \ldots, n$, be arbitrary non-decreasing penalty functions in the problem $1 \mid r_j \mid \varphi_{\max}$. Then we have $\nu_n \geq \nu_k$ for all $k = 1, 2, \ldots, n$, i.e., $\nu^* = \nu_n$.*

Proof. Suppose that there exists a number k, $k < n$, such that $\nu_k > \nu_n$. Assume that $\pi_n = (j_1, j_2, \ldots, j_n)$ is a schedule for which the value ν_n is obtained. Then we consider the schedule

$$\pi = (j_{n-k+1}, j_{n-k+2}, \ldots, j_n, j_1, j_2, \ldots, j_{n-k}).$$

Please note that under the schedule π, the job j_n will be carried out as the k-th job in the sequence. Since $C_{j_n}(\pi_n) \geq C_{j_n}(\pi)$, we have

$$\nu_n = \varphi_{j_n}(C_{j_n}(\pi_n)) \geq \varphi_{j_n}(C_{j_n}(\pi)) \geq \nu_k.$$

Due to the assumption $\nu_k > \nu_n$, we obtain the inequality

$$\nu_k > \nu_n \geq \nu_k$$

which is a contradiction. The lemma has been proved. □

Thus, the solution of the dual problem $1|r_j|\phi_{max}$ is reduced to the problem of finding the value ν_n. We enumerate all jobs in order of non-decreasing release dates: $r_{i_1} \le r_{i_2} \le \cdots \le r_{i_n}$. Due to

$$\nu_n = \min_{\pi \in \Pi(N)} \varphi_{j_n}(C_{j_n}(\pi)),$$

we will put each of the jobs j of the set N onto the last (i.e., the n-th) position. The other $n-1$ jobs of the set $N \setminus \{j\}$ are arranged in their original order starting at time τ. This gives the earliest completion time of processing the jobs from the set $N \setminus \{j\}$. This procedure is formally summarized in Algorithm 1. The input data of Algorithm 1 is the pair (N, τ), where τ is used to calculate C_j.

Algorithm 1: Solution of the dual problem of the problem $1 \mid r_j \mid \varphi_{\max}$

1. Construct the schedule $\pi^r = (i_1, i_2, \ldots, i_n)$, in which all jobs are sequenced according to non-decreasing release dates: $r_{i_1} \le r_{i_2} \le \cdots \le r_{i_n}$.
2. For $k = 1, 2, \ldots, n$, find the value $\varphi_{i_k}(C_{i_k}(\pi_k))$ for the schedule $\pi_k = (\pi^r \setminus i_k, i_k)$.
3. Find the value $\nu^* = \min\limits_{k=1,\ldots,n} \varphi_{i_k}(C_{i_k}(\pi_k))$ and the job i_k, which gives the value ν^*.

The complexity of Algorithm 1 can be estimated as follows. We need $O(n \log n)$ operations to construct the schedule π^r. We need $O(n)$ operations to find each of the n values $\varphi_{i_k}(C_{i_k}(\pi_k))$. Therefore, to determine the value ν^* and the corresponding job i_k, for which the value ν^* is obtained, no more than $O(n^2)$ operations are required.

Theorem 1 ([25]). *Let $\varphi_j(t), j = 1, 2, \ldots, n$, be arbitrary non-decreasing penalty functions for the problem $1 \mid r_j \mid \varphi_{\max}$. Then the inequality $\mu^* \ge \nu^*$ holds.*

Proof. Suppose the opposite, i.e., there exists an instance of the problem $1 \mid r_j \mid \varphi_{\max}$, for which the inequality $\mu^* < \nu^*$ holds. Let $\pi^* = (j_1, j_2, \ldots, j_n)$ be an optimal schedule for this instance. Then we have

$$\varphi_{j_n}(C_{j_n}(\pi^*)) \le \mu^* < \nu^*,$$

which contradicts equality (3):

$$\nu^* = \nu_n = \min_{\pi \in \Pi(N)} \varphi_{j_n}(C_{j_n}(\pi)).$$

The theorem has been proved. □

The obtained estimate can be efficiently used in constructing schemes of a branch and bound method for solving the problem $1 \mid r_j \mid \varphi_{\max}$, and for estimating the error of approximate solutions when the branch and bound algorithm stops without finding an optimal solution.

We denote by $\{N', \tau', \nu', \pi', B'\}$ the sub-problem of processing the jobs of the set $N' \subseteq N$ from time $\tau' \ge \tau$ on according to some partial sequence π' for the jobs of the set $N \setminus N'$, where ν' is the lower bound obtained by solving the dual problem of this instance, $\tau' = C_{\max}(\pi', \tau)$ is the start time of the planning horizon for the jobs from the set B', which is equal to the makespan value for the sequence π', and τ is the time when the machine is ready to process the jobs from the set N. B' is the set of jobs that cannot be placed on the corresponding first position of the schedule.

The subsequent Algorithm 2 implements one of the possible schemes of the branch and bound method using the solution of the dual problem. The branching in Algorithm 2 is carried out as a result of dividing the current sub-instance into two instances: put the job f (the job with the smallest due date from the set of jobs ready for processing) at the next position in the schedule and prohibit the inclusion of the job f at the next position in the schedule (by increasing the possible start time of job f).

Denote

$$r_j(\tau) = \max\{r_j, \tau\}, \quad r(N, \tau) = \min_{j \in N} r_j(\tau).$$

Let $|N| > 1$. We choose a job $f = f(N, \tau)$ from the set N such that

$$f(N, \tau) = \arg \min_{j \in N \setminus B} \Big\{ d_j \mid r_j(\tau) = r(N, \tau) \Big\}.$$

If $N = \{i\}$, then $f(N, \tau) = i$ for all τ. Here B is a set of jobs that cannot be placed on the current first position. Denote by π^* the currently best schedule constructed for all jobs.

We note that a deeper comparative discussion of the characteristics of the particular search strategies can be found in [26,27].

Algorithm 2: Solution of the problem $1 \mid r_j \mid L_{\max}$ by the branch and bound method based on the solution of the dual problem

---

1. Initial step

Let $\pi^* = \emptyset$. The list of instances contains the original instance $\{N, \tau, \nu, \emptyset, \emptyset\}$, where ν is a lower bound on the optimal objective function value obtained by solving the dual problem of this instance.

2. Main step

(a) From the list of instances, select an instance $\{N', \tau', \nu', \pi', B'\}$ with a minimal lower bound ν'.

(b) Find the job $f = f(N', \tau')$ from the set $N' \setminus B'$ with the smallest due date from the number of jobs ready for processing at time τ'.

(c) Replace the instance $\{N', \tau', \nu', \pi', B'\}$ by the two instances $\{N_1, \tau_1, \nu_1, \pi_1, B_1\}$ and $\{N_2, \tau_2, \nu_2, \pi_2, B_2\}$ in the list of instances, where:

- $N_1 = N' \setminus \{f\},\ \tau_1 = \max\{r_f, \tau'\} + p_f,\ B_1 = \emptyset,\ \pi_1 = (\pi', f),\ \nu_1$ is a lower bound obtained by solving the dual problem of this instance by Algorithm 3;
- $N_2 = N',\ \tau_2 = \tau',\ B_2 = B' \cup \{f\},\ \pi_2 = \pi',\ \nu_2$ is a lower bound obtained by solving the dual problem of this instance by Algorithm 3.

(d) If, after completing this step of the algorithm, we obtain $\{\pi_1\} = N$, that is, all jobs are ordered, then $\pi^* = \arg\min\{L_{\max}(\pi_1, \tau), L_{\max}(\pi^*, \tau)\}$.

(e) Exclude all instances $\{N', \tau', \nu', \pi', B'\}$, for which $\nu' \geq L_{\max}(\pi^*, \tau)$.

3. Termination step

If the list of instances is empty, STOP, otherwise repeat the main step 2.

To find the value ν in step 2(c), we need to modify Algorithm 1 taking into account a list B of jobs that cannot be placed on the current position. The input data of Algorithm 3 is the triplet (N, τ, B), where τ is used to calculate C_j.

Algorithm 3: Modification of the solution algorithm for the dual problem of the problem $1 \mid r_j \mid \varphi_{\max}$ with respect to a list of jobs that cannot be on the first position

1. Construct the schedule $\pi^r = (i_1, i_2, \ldots, i_n)$, in which all jobs are sequenced according to non-decreasing release dates: $r_{i_1} \le r_{i_2} \le \cdots \le r_{i_n}$.
2. For $k = 1, 2, \ldots, n$, if $N \setminus (B \cup \{i_k\}) \neq \emptyset$, find a job

$$i_l = \arg \min_{j \in N \setminus (B \cup \{i_k\})} r_j$$

and construct the schedule $\pi_k = (i_l, \pi^r \setminus \{i_l, i_k\}, i_k)$.
Find the value $\varphi_{i_k}(C_{i_k}(\pi_k))$.
If $N \setminus (B \cup \{i_k\}) = \emptyset$, then we assume $C_{i_k} = +\infty$.
3. Find the value $\nu^* = \min_{k=1,\ldots,n} \varphi_{i_k}(C_{i_k}(\pi_k))$ and the job i_k, which gives the value ν^*.

It is easy to see that this algorithm can be used to solve the more general problem $1 \mid r_j \mid \varphi_{\max}$. In addition, if the algorithm is stopped without an empty list of instances due to a time limit, the current schedule π^* can be taken as an approximate solution of the problem.

Hence, although the original problem $1 \mid r_j \mid \varphi_{\max}$ is NP-hard in the strong sense (recall that problem $1 \mid r_j \mid L_{\max}$ is NP–hard in the strong sense), the dual problem turned out to be polynomially solvable.

If precedence relations are specified between the jobs by an acyclic graph G, then the dual problem of the problem $1 \mid r_j, prec \mid \varphi_{\max}$ can also be solved in a similar way. Since the argumentation is similar, we skip the details. Here, the core is to solve the problem $1 \mid r_j, prec \mid C_{\max}$. Jobs without successors according to the precedence graph G will be put one-by-one to the last positions in the job sequence. Thus, the dual problem of the problem $1 \mid r_j, prec \mid \varphi_{\max}$ is also polynomially solvable.

For problems with $m > 1$ machines, e.g., problem $Pm \mid r_j, prec \mid \varphi_{\max}$, the core consists of solving the dual problem, which is the partition problem. This dual problem is NP–hard in the ordinary sense.

Thus, although in mathematical programming the original and dual problems have usually the same complexity status, it turned out that the dual problems of the scheduling problems considered in this paper have a lower complexity than the original problems. This interesting fact should be investigated further in more detail also for other scheduling problems.

3. The Inverse Problem of the Maximum Lateness Problem

The inverse problem of the NP-hard problem of minimizing maximum lateness $1 \mid r_j \mid L_{\max}$ consists of finding a schedule π, which reaches the maximum minimal lateness and finding the value

$$\lambda^* = \max_{\pi \in \Pi(N)} \min_{k=1,\ldots,n} L_{j_k}(C_{j_k}(\pi)). \tag{5}$$

Please note that for this problem, inserted idle times of the machine are prohibited.

This problem was solved only for the case of simultaneous availability of the set N for processing, i.e., $r_j = 0$, for all $j \in N$ in [28]. We consider the general case of the problem $1 \mid r_j \mid \max L_{\min}$.

Lemma 2. *There exists an optimal schedule $\pi = (i_1, \ldots, i_n)$ for the problem $1 \mid r_j \mid \max L_{\min}$, for which*

$$d_{i_k} - p_{i_k} \le d_{i_{k+1}} - p_{i_{k+1}},\ k = 2, 3, \ldots, n-1, \tag{6}$$

and

$$\lambda^* = \min_{k=1,\ldots,n} L_{i_k}(C_{i_k}(\pi)).$$

Proof. Assume that at least one of inequalities (6) is not satisfied for an optimal schedule $\pi' = (j_1, \ldots, j_n)$ and let

$$\lambda^* = \min_{k=1,\ldots,n} L_{i_k}(C_{i_k}(\pi')).$$

In what follows, the proof will consist of two stages, which can be repeated several times.

Step 1. If there are no machine idle times in the schedule π', then go to step 2. Let there be idle times according to schedule π', and consider the last of them:

$$C_{j_k} < r_{j_{k+1}} \qquad \text{and} \quad r_{j_m} \le C_{j_{m-1}}, \quad m = k+2, \ldots, n.$$

Construct the schedule $\pi'' = (j_{k+1}, j_1, \ldots, j_k, j_{k+2}, \ldots, j_n)$. Since

$$C_j(\pi'') \ge C_j(\pi') \quad \text{for all } j \in N,$$

the value of the minimal lateness will not decrease. There will be no idle time under the schedule π'', and the optimal value λ^* will be saved. Set $\pi' := \pi''$ and go to step 2.

Step 2. If the schedule π' meets the conditions of Lemma 2, the proof is completed. If there exist two jobs j_l, j_{l+1}, for which

$$d_{j_l} - p_{j_l} > d_{j_{l+1}} - p_{j_{l+1}},$$

then exchange the jobs j_l, j_{l+1} which yields the schedule

$$\pi'' = (j_1, \ldots, j_{l-1}, j_{l+1}, j_l, j_{l+2}, \ldots, j_n).$$

As there are no machine idle times under the schedule π', we have

$$r_{j_l} \le C_{j_{l-1}}(\pi').$$

There are the following possible cases:

(1) Let $r_{j_{l+1}} \le C_{j_{l-1}}(\pi')$. Obviously, in this case we have

$$C_{j_k}(\pi') = C_{j_k}(\pi''), \quad k = 1, 2, \ldots, l-1, l+2, \ldots, n. \tag{7}$$

According to the assumptions, inequality

$$C_{j_{l-1}}(\pi') + p_{j_l} + p_{j_{l+1}} - d_{j_{l+1}} > C_{j_{l-1}}(\pi') + p_{j_l} - d_{j_l}. \tag{8}$$

holds. Moreover, we have

$$C_{j_{l-1}}(\pi') + p_{j_{l+1}} - d_{j_{l+1}} > C_{j_{l-1}}(\pi') + p_{j_l} - d_{j_l}; \tag{9}$$

$$C_{j_{l-1}}(\pi') + p_{j_{l+1}} - d_{j_l} > C_{j_{l-1}}(\pi') + p_{j_l} - d_{j_l}. \tag{10}$$

Formulas (7)–(10) show that the maximal lateness is not reduced. Set $\pi' := \pi''$ and repeat step 2.

(2) Let $r_{j_{l+1}} > C_{j_{l-1}}(\pi')$. In this case, we have

$$C_{j_k}(\pi'') = C_{j_k}(\pi'), \quad k = 1, 2, \ldots, l-1, \tag{11}$$

$$C_{j_k}(\pi'') > C_{j_k}(\pi'), \quad k = l+2, \ldots, n. \tag{12}$$

According to the assumptions, we have

$$C_{j_{l+1}}(\pi'') - d_{j_{l+1}} > C_{j_l}(\pi') - d_{j_l}; \tag{13}$$

$$C_{j_l}(\pi'') - d_{j_l} > C_{j_{l+1}}(\pi') - d_{j_{l+1}}. \tag{14}$$

Formulas (8) and (11)–(14) show that the maximal lateness is not reduced. Set $\pi' := \pi''$ and go to step 1.

In a finite number of steps, we construct an optimal schedule satisfying the conditions of the lemma. The lemma has been proved. □

Algorithm 4 constructs n schedules, one of which is satisfying the conditions of the lemma.

Algorithm 4: Solution of the inverse problem $1 \mid r_j \mid \max L_{\min}$

1. Enumerate all jobs of the set N according to $d_1 - p_1 \le d_2 - p_2 \le \cdots \le d_n - p_n$.
2. For $k = 1, 2, \ldots, n$ do:
   (a) construct the schedule $\pi_k = (k, 1, \ldots, k-1, k+1, \ldots, n)$ and
   (b) determine $\lambda_k = \min\limits_{j=1,\ldots,n} L_j(C_j(\pi_k))$.
3. Calculate $\lambda^* = \max\limits_{k=1,\ldots,n} \lambda_k$.

$O(n \log n)$ operations are needed for renumbering the jobs of the set N. $O(n)$ operations are needed for constructing the schedule π_k and calculating the value λ_k, $k = 1, \ldots, n$. Thus, no more than $O(n^2)$ operations are needed to find the value λ^*.

The objective function value of a solution of the problem of maximizing minimal lateness $1 \mid r_j \mid \max L_{\min}$ is a lower bound on the optimal objective function value for the original problem $1 \mid r_j \mid L_{\max}$.

Theorem 2 ([18]). *For the optimal function values of the problem* $1 \mid r_j \mid L_{\max}$ *and the corresponding inverse problem* $1 \mid r_j \mid \max L_{\min}$, *the inequality* $\mu^* \ge \lambda^*$ *holds.*

Proof. Denote by π' and π'' optimal schedules for the problems $1 \mid r_j \mid L_{\max}$ and $1 \mid r_j \mid \max L_{\min}$, respectively. There exist jobs $k', k'' \in N$, for which the following inequalities hold:

$$\mu^* = C_{k'}(\pi') - d_{k'} \ge C_j(\pi') - d_j \quad \text{for all } j \in N; \tag{15}$$

$$C_j(\pi'') - d_j \ge C_{k''}(\pi'') - d_{k''} = \lambda^* \quad \text{for all } j \in N. \tag{16}$$

Please note that jobs k' and k'' can be identical. Let $\pi'' = (j_1, \ldots, j_n)$. Obviously, the following inequality is satisfied for job j_1:

$$C_{j_1}(\pi') - d_{j_1} \ge C_{j_1}(\pi'') - d_{j_1}. \tag{17}$$

According to inequalities (15)–(17), we get

$$\mu^* = C_{k'}(\pi') - d_{k'} \ge C_{j_1}(\pi') - d_{j_1} \ge C_{j_1}(\pi'') - d_{j_1} \ge C_{k''}(\pi'') - d_{k''} = \lambda^*,$$

i.e., $\mu^* \ge \lambda^*$. The theorem has been proved. □

4. Computational Results

In this section, we present some results of the numerical experiments carried out on randomly generated test instances. The numerical experiments were carried out on a PC Intel® Core™ i5-4210U CPU @ 1.70GHz, 4 cores; 8 GB DDR4 RAM.

Various methods of generating test instances for different types of scheduling problems are described in [29]. For the problem $1|r_j|L_{max}$ with n jobs, the authors suggest the following generation scheme:

- $r_0 = 0$ and $r_i = r_{i-1} + X_i$, where $X_i \sim \exp(\lambda)$, for $i = 1, ..., n$;
- $p_i \sim N(\mu, \sigma)$ for $i = 1, \ldots, n$, truncated below a known lower bound;
- $d_i = r_i + kE[p_i]$ for $i = 1, \ldots, n$, where $k \geq 1$.

The authors suggest that λ, μ, σ and k are generation parameters that can be fixed by the user. Applying this generation scheme, we generate release dates which are independent from the processing times, while the due dates correlate with the processing times, as it usually happens in real problems. We set

$$\lambda = \frac{1}{100}, \quad \mu = 100, \quad \sigma = 40, \quad k = 1$$

and generated 15 instances for each $n \in \{3, 4, ..., 20\}$. The results are shown in Table 1, where the number of branching points are given for each of the 15 instances for any value of n.

Table 1. Number of branching points in Algorithm 2 for the test instances generated according to [29].

n	Number of the Instance														
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
3	4	2	2	2	2	2	2	2	2	2	2	4	2	2	2
4	3	3	3	3	3	3	3	3	3	5	4	3	3	3	3
5	9	4	4	25	4	4	31	4	4	8	4	4	4	4	18
6	5	5	64	227	161	13	7	5	5	5	5	5	84	15	5
7	6	25	16	6	6	6	31	147	310	34	35	6	172	6	222
8	7	234	33	7	7	23	7	391	7	7	7	7	194	7	7
9	1271	8	1847	8	3786	2934	8	8	62	8	8	721	8	208	8
10	9	690	9	9	66	114	9	9	9	9	9	9	3944	6584	9
11	10	13,098	10	5071	10	13,992	11	10	6081	773	591	303	10	3772	10
12	11	22,736	11	11	35,910	57,407	32,798	11	139	164,613	11	11	166	11	207,119
13	12	12	213,433	1109	*	12	183	227	794,567	12	*	12	30,946	*	12
14	13	13	458	*	5933	21,774	4394	13	13	13	*	32,246	*	13	*
15	2293	14	*	*	*	*	14	*	14	14	*	*	14	14	*
16	15	*	15	*	15	*	15	*	15	*	*	*	172,876	15	*
17	*	16	16	*	16	*	16	16	16	16	*	9128	*	16	16
18	17	*	*	17	*	*	17	17	17	17	*	*	*	*	*
19	*	18	18	*	*	*	*	*	4999	18	18	*	*	18	18
20	*	19	2103	*	*	19	20	19	19	19	19	19	19	19	*

An asterisk (*) in Table 1 means that the solution could not be found within 15 min. According to this table, a large part of the instances can be solved with several branching points not greater than the number of jobs. However, some instances appeared to be hard and required a much larger number of branching points. It can be observed that these hard instances generated according to [29] display a rather different solution behavior and need very different numbers of branching points for their exact solution. This interesting phenomenon deserves further detailed investigations which are planned as future work by the authors.

Next, we consider an instance of the problem as a point in the $3n$-dimensional space, where each value of r_i, p_i, d_i represents one of the dimensions. We consider the vector from the zero point to the point of the instance. The complexity of an instance is defined by the direction of the vector, but not by its length. Therefore, to explore the complexity of different instances, we can take points on the surface of the $3n$-dimensional cube. For the processing times and the release dates, we consider only non-negative values. As a result, we obtain points on the quarter of the surface of the $3n$-dimensional cube with $r_i \geq 0$ and $p_i \geq 0$. Let the size of the cube be 100. We generated 300,000 points on the surface for problems with 4, 5, 6, 7, 8 and 9 jobs (i.e., in the 12-, 15-, 18-, 21, 24, 27-dimensional spaces, respectively). All instances have been solved, and we counted the number of iterations (i.e., branching points) for each problem instance. The results are shown in Figure 1 for the instances with 4 jobs,

in Figure 2 for the instances with 5 jobs, in Figure 3 for the instances with 6 jobs, in Figure 4 for the instances with 7 jobs, in Figure 5 for the instances with 8 jobs and in Figure 6 for the instances with 9 jobs. At the y-axis, the numbers are given how often a particular number of branching points (Figure 1) or an interval for the number of branching points (Figures 2–6) has occurred among the 300,000 instances for each number n of jobs. For example, in Figure 1, one can see that among the 300,000 solved instances with 5 jobs, there were 72,897 instances with 3 branching points, 16,131 instances with 4 branching points, 29,342 instances with 5 branching points, and so on. It can be observed that the maximal number of 20 branching points was reached for 1294 instances, which is approximately equal to 0.4% of all instances. In Figure 2, the numbers of branching points are grouped in intervals of 5 in each column, i.e., there were 86 026 instances with several branching points between 0 and 4 (actually 4, because it is the minimum possible number of branching points for instances with 5 jobs), 28,566 instances with several branching points between 5 and 9, etc. For instances with 5 jobs, the maximum number of branching points was 93, and it was reached for only 33 instances. As it can be seen in the figures, most of instances can be solved by a small number of branches. For a larger number of jobs, one can detect a smaller number of hard instances with a large number of branching points. Thus, for the instances with 9 jobs, among the 300,000 solved instances, there were only two instances with several branching points more than 180,000: one with 184,868, and the other with 191,887.

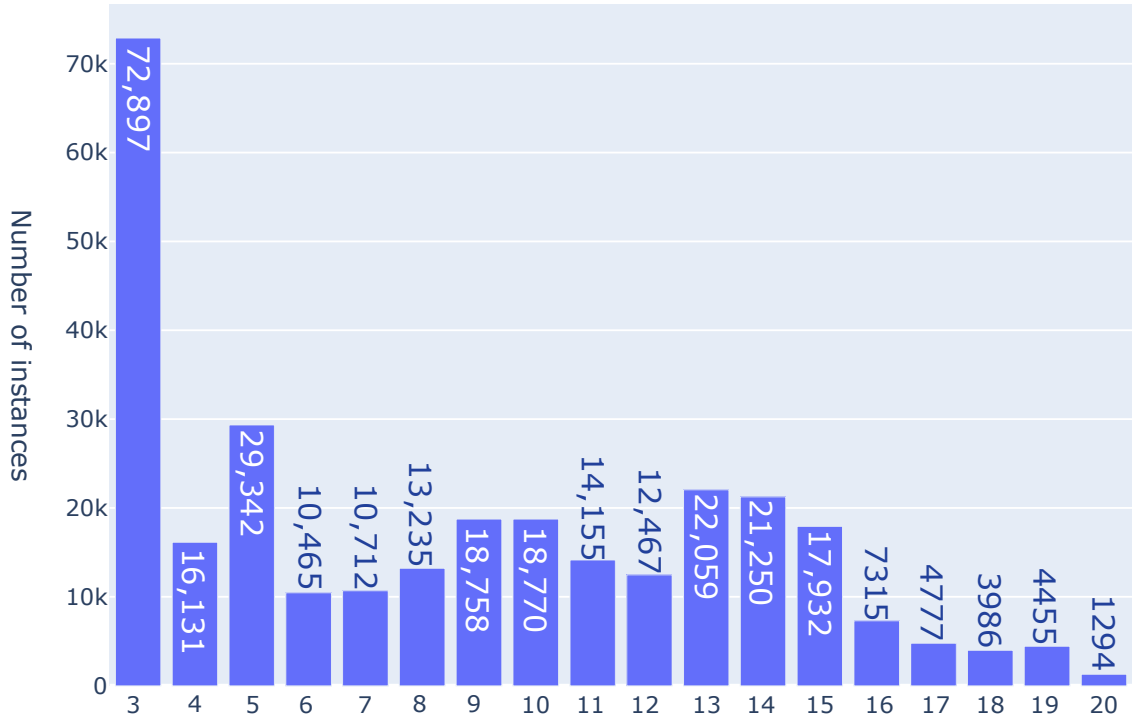

Figure 1. Number of branching points for the instances with 4 jobs.

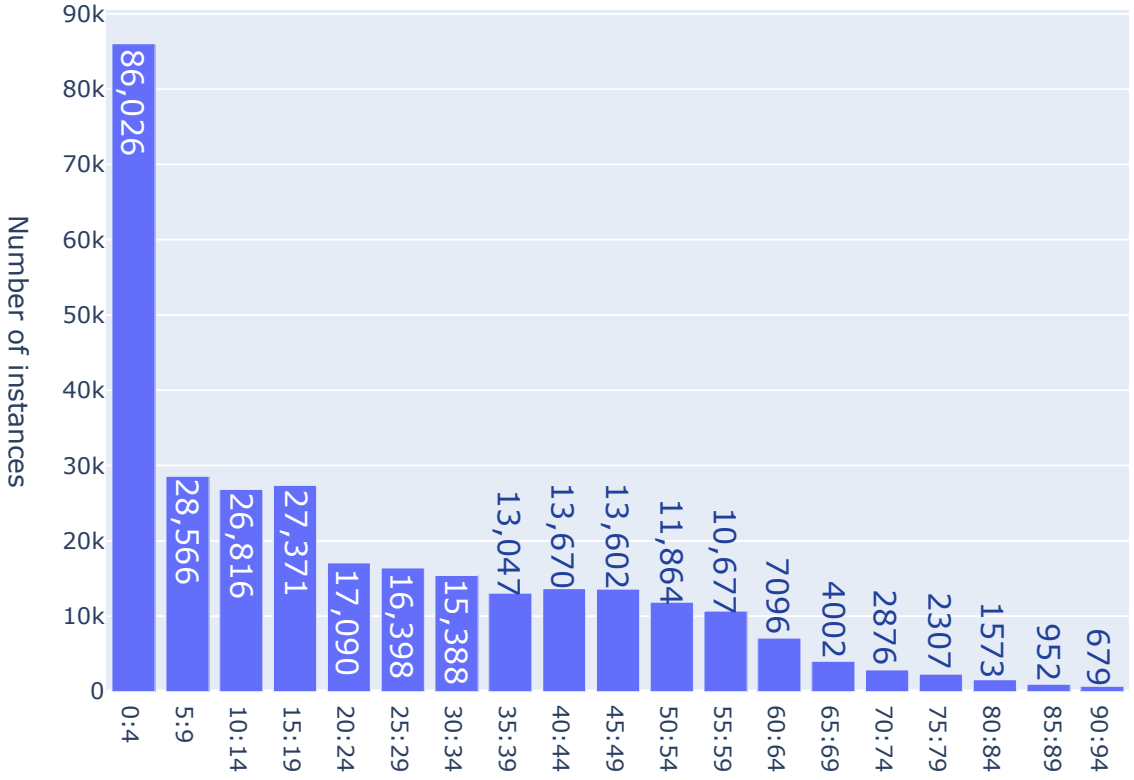

Figure 2. Number of branching points for the instances with 5 jobs.

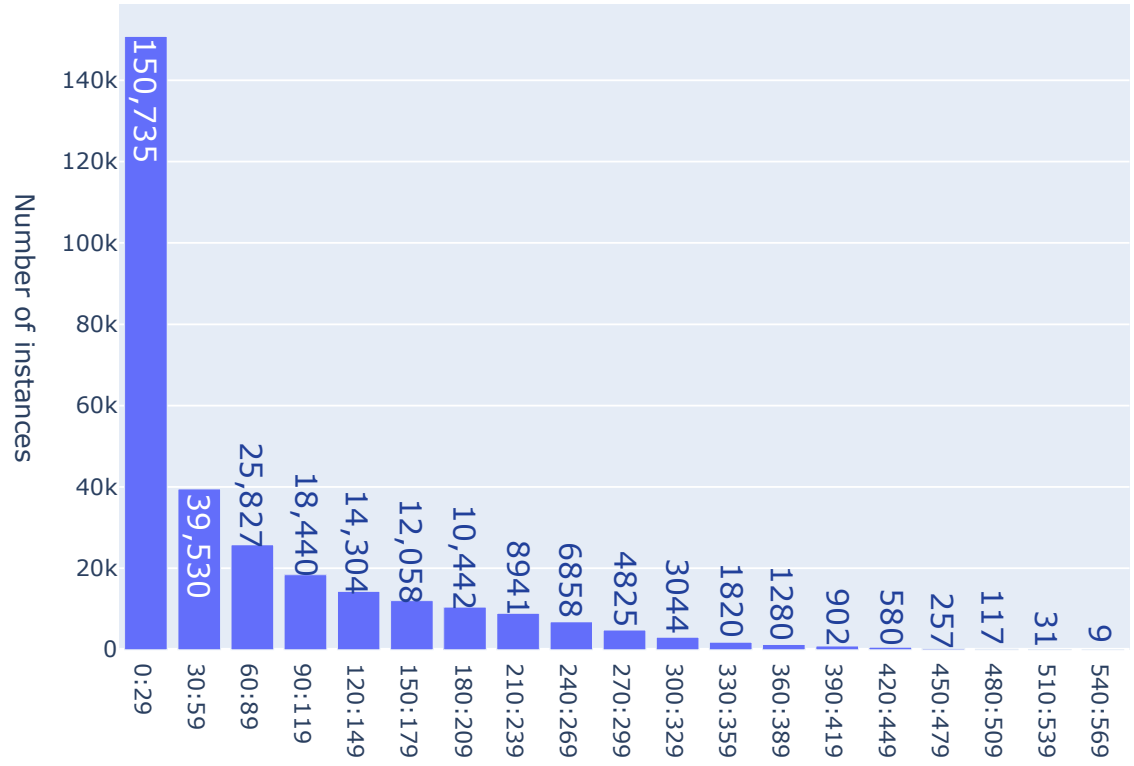

Figure 3. Number of branching points for the instances with 6 jobs.

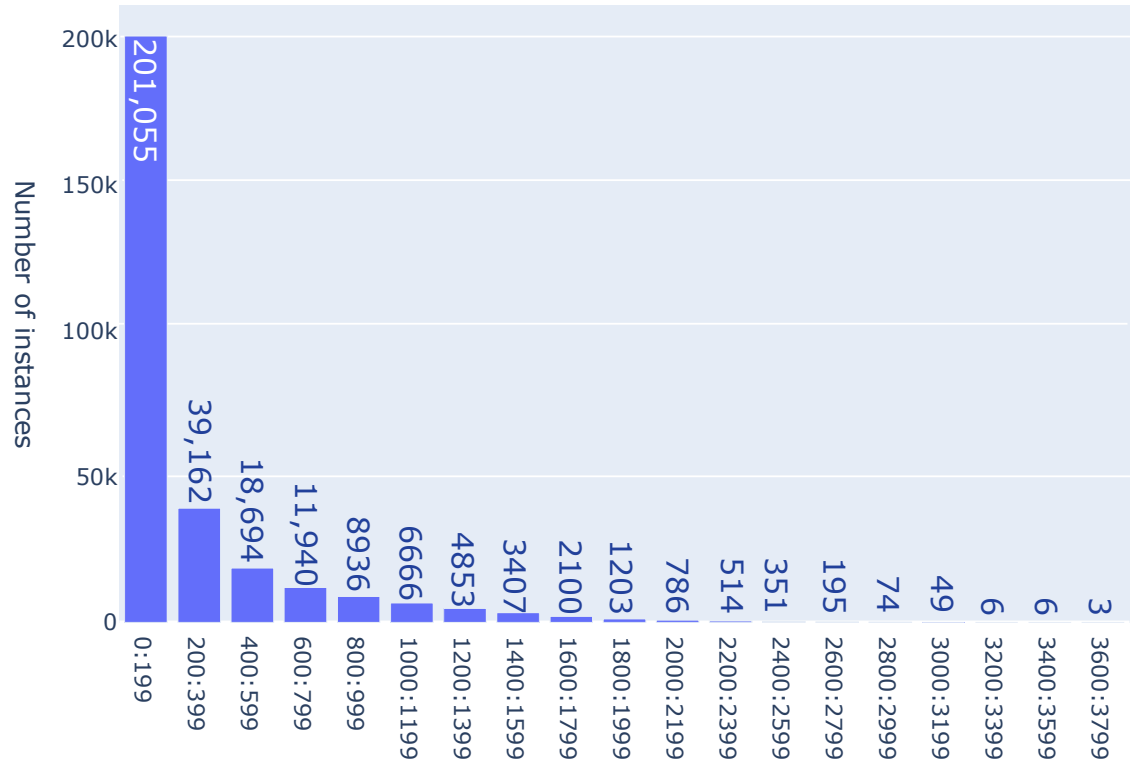

Figure 4. Number of branching points for the instances with 7 jobs.

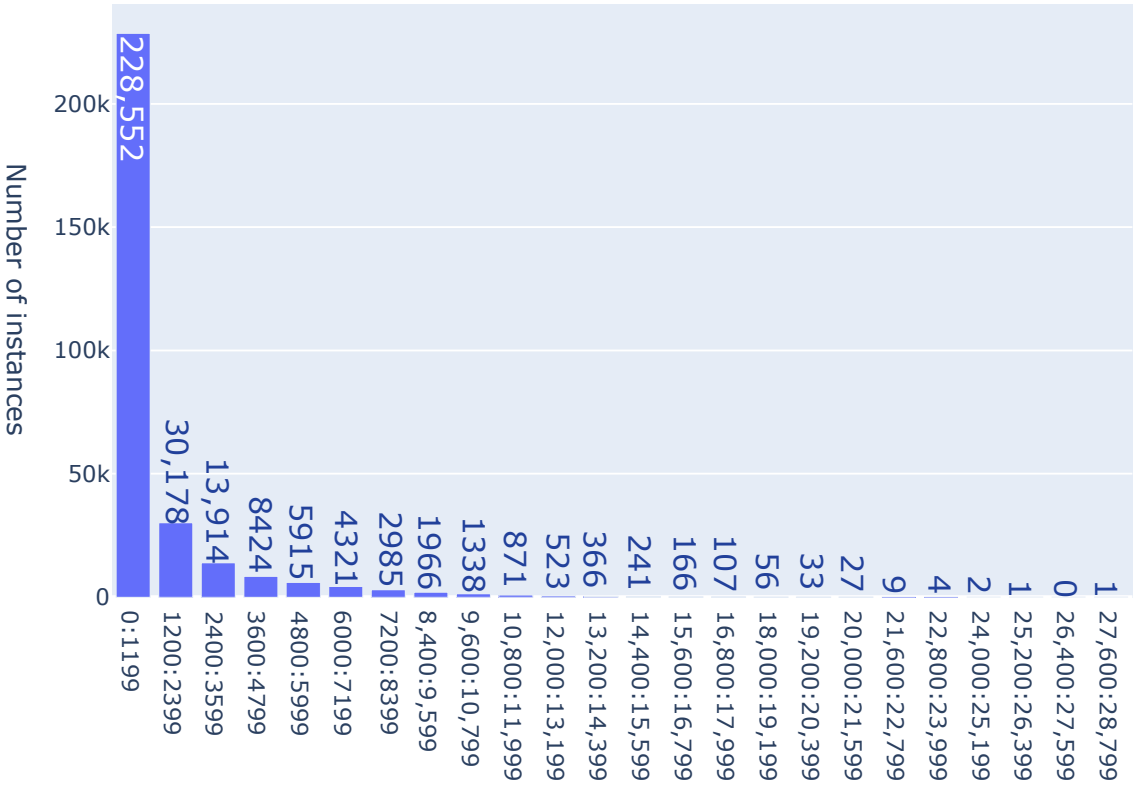

Figure 5. Number of branching points for the instances with 8 jobs.

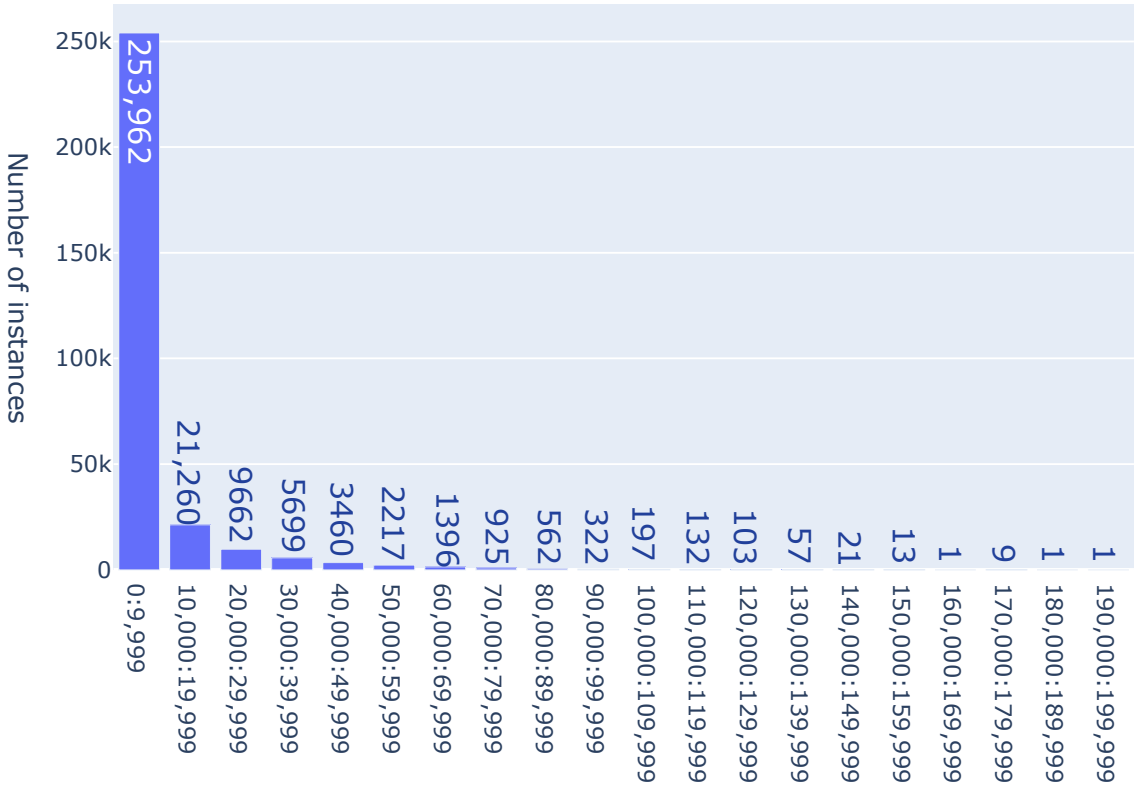

Figure 6. Number of branching points for the instances with 9 jobs.

5. Conclusions

For solving the NP-hard problem $1 \mid r_j \mid \varphi_{\max}$ for arbitrary non-decreasing penalty functions, an algorithm has been proposed which implements the branch and bound method. For each sub-instance to be considered, a lower bound on the optimal function value is determined using a solution of the dual problem. The proposed algorithm for solving the dual problem can find a solution in several operations not exceeding $O(n^2)$. The proposed algorithm can find an optimal solution within a time limit of one second for about 98% of the instances for 8 jobs and for about 85% of instances for 9 jobs. Although there are a few instances with a large number of branching points, most instances can be solved very fast by the proposed algorithm. For the hard instances, the execution of the algorithm can be interrupted at any moment, and the current objective function value with the corresponding schedule π can be used as an approximate solution for the instance. However, some generated instances appeared to be very hard. At the moment, we cannot explain this interesting phenomenon. It requires deep additional investigations which are planned in the future.

In addition to the dual problem, the inverse problem has also been solved for the lateness objective function. The algorithm for solving the inverse problem has a complexity of $O(n^2)$ operations. However, in the problem of minimizing maximum lateness, one tries to 'equalize' the lateness while minimizing the maximum, in the inverse problem the lateness values are 'equalized' due to the maximization of the minimum provided that inserted machine idle times are prohibited.

Author Contributions: The results described in this paper were obtained by communications and mutual works of the authors. Conceptualization, A.A.L. and N.P.; data curation, A.A.L. and N.P.; formal analysis, A.A.L., N.P. and F.W.; investigation, A.A.L., N.P. and F.W.; methodology, A.A.L., N.P. and F.W.; project administration, A.A.L.; software, N.P.; supervision, A.A.L. and F.W.; validation, A.A.L., N.P. and F.W.; visualization, N.P.; writing—original draft, A.A.L. and N.P.; writing—review & editing, F.W. All authors have read and agreed to the published version of the manuscript.

Funding: The research was partially supported by RFBR project 18-07-00656 and partially supported by the Basic Research Program of the National Research University Higher School of Economics.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lenstra, J.K.; Rinnooy Kan, A.H.G.; Brucker, P. Complexity of Machine Scheduling Problems. *Ann. Discret. Math.* **1977**, *1*, 343–362.
2. Brooks, G.N.; White, C.R. An Algorithm for Finding Optimal or Near-optimal Solutions to the Production Scheduling Problem. *J. Ind. Eng.* **1965**, *16*, 34–40.
3. Schrage, L. Solving Resource-Constrained Network Problems by Implicit Enumeration: Non Preemptive Case. *Oper. Res.* **1970**, *18*, 263–278. [CrossRef]
4. Burduk, V.Y.; Shkurba, V.V. Scheduling Theory. Problems and Solving Methods. *Cybernetics* **1971**, *1*, 89–102. (In Russian)
5. McMahon, G.; Florian, M. On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness. *Oper. Res.* **1975**, *23*, 475–482. [CrossRef]
6. Korbut, A.A.; Sigal, I.H.; Finkelshtein, Y.Y. Branch and Bound Method (Survey of Theory, Algorithms, Programs and Applications). *Oper. Forsch. Stat. Ser. Optim.* **1977**, *8*, 253–280. (In Russian)
7. Carlier, J. The One-Machine Sequencing Problem. *Eur. J. Oper. Res.* **1982**, *11*, 42–47. [CrossRef]
8. Tanaev, V.S.; Gordon, V.S.; Shafranskiy, Y.M. *Scheduling Theory. Single-Stage Systems Scheduling Theory*; Springer-Science+Business Media, B.V.: Dordrecht, The Netherland, 1994; 374p.
9. Hoogeveen, J. Minimizing Maximum Promptness and Maximum Lateness on a Single Machine. *Math. Oper. Res.* **1996**, *21*, 100–114. [CrossRef]
10. Vakhania, N.; Werner, F. Minimizing Maximum Lateness of Jobs with Naturally Bounded Job Data on a Single Machine. *Theor. Comput. Sci.* **2013**, *501*, 72–81. [CrossRef]
11. Lageweg, B.; Lenstra, J.; Rinnooy Kan, A. Minimizing Maximum Lateness on one machine: Computational experience and applications. *Stat. Neerl.* **1976**, *30*, 25–41. [CrossRef]
12. Liu, Z. Single Machine Scheduling to Minimize Maximum Lateness Subject to Release Dates and Precendence Constraints. *Comp. Oper. Res.* **2010**, *37*, 1537–1543. [CrossRef]
13. Kacem, I.; Kellerer, H. Approximation Schemes for Minimizing Maximum Lateness. *Algorithmica* **2018**, *80*, 3825–3843. [CrossRef]
14. Wang, D.; Xu, D. Single-Machine Scheduling with Workload Dependent Maintenance Duration to Minimize Maximum Lateness. *Math. Probl. Eng.* **2015**, *2015*, 732437. [CrossRef]
15. Sels, V.; Vanhoucke, M. A Hybrid Elektromagnetism-like Mechanism/Tabu Search Procedure for the Single Machine Scheduling Problem with a Maximum Lateness Objective. *Comp. Ind. Eng.* **2014**, *67*, 44–55. [CrossRef]
16. Lazarev, A.A.; Lemtyuznikova, D.; Werner, F. *A General Approximation Approach for Multi-Machine Problems with Minimizing the Maximum penalty*; Preprint 04/19, FMA; OVGU: Magdeburg, Germany, 2019; 25p.
17. Vakhania, N. Scheduling a Single Machine with Primary and Secondary Objectives. *Algorithms* **2018**, *11*, 80. [CrossRef]
18. Lazarev, A.A. *Scheduling Theory. Methods and Algorithms*; ICS RAS: Moscow, Russia, 2019; 408p. (In Russian)
19. Lazarev, A.A. Pareto-optimal set for the NP-hard problem of minimization maximum lateness. Izvestiya of Russian Academy of Sciences. *Theory Syst. Control* **2016**, *6*, 103–110. (In Russian)
20. Lazarev, A.A.; Arkhipov, D.; Werner, F. Scheduling with Equal Processing Times on a Single Machine: Minimizing Maximum Lateness and Makespan. *Optim. Lett.* **2017**, *11*, 165–177. [CrossRef]
21. Aloulou, M.A.; Kovalyov; M.Y.; Portmann, M.-C. Maximization Problems in Single Machine Scheduling. *RAIRO Oper. Res.* **2007**, *41*, 1–18. [CrossRef]
22. Gafarov, E.R.; Lazarev, A.A.; Werner, F. Algorithms for Some Maximization Scheduling Problems on a Single Machine. *Autom. Remote. Control* **2010**, *71*, 2070–2084. [CrossRef]
23. Gafarov, E.R.; Lazarev, A.A.; Werner, F. Single machine Total Tardiness Maximization Problems. *Ann. Oper. Res.* **2013**, *207*, 121–136. [CrossRef]
24. Gafarov, E.R.; Lazarev, A.A.; Werner, F. Transforming a Pseudo-polynomial Algorithm for the Single Machine Total Tardiness Maximization Problem into a Polynomial One. *Ann. Oper. Res.* **2012**, *196*, 247–261. [CrossRef]

25. Lazarev, A.A. Dual of the Maximum Cost Minimization Problem. *J. Sov. Math.* **1989**, *44*, 642–644.
26. Agin, N. Optimum Seeking with Branch and Bounds. *Manag. Sci.* **1966**, *13*, 176–185. [CrossRef]
27. Lawler, E.L.; Wood, D.E. Branch and Bounds Methods: A Survey. *Oper. Res.* **1966**, *14*, 699–719.[CrossRef]
28. Conway, R.W.; Maxwell, W.L.; Miller, L.W. *Theory of Scheduling*; Addison-Wesley: Reading, MA, USA, 1967.
29. Hall, N.G.; Posner, M.E. Generating Experimental Data for Computational Testing with Machine Scheduling Applications. *Oper. Res.* **2001**, *49*, 854–865. [CrossRef]