Contents lists available at ScienceDirect

Mathematical and Computer Modelling

journal homepage: www.elsevier.com/locate/mcm

Minimizing total weighted flow time of a set of jobs with interval processing times

Yu.N. Sotskov[a,*], N.G. Egorova[a], T.-C. Lai[b]

[a] United Institute of Informatics Problems, Surganova Street 6, Minsk 220012, Belarus

[b] National Taiwan University, Sec. 4, Roosevelt Road 85, Taipei 106, Taiwan

ARTICLE INFO

Article history:
Received 2 December 2008
Received in revised form 19 January 2009
Accepted 3 March 2009

Keywords:
Single-machine scheduling
Uncertainty
Domination

ABSTRACT

We consider a single-machine scheduling problem with each job having an uncertain processing time which can take any real value within each corresponding closed interval before completing the job. The scheduling objective is to minimize the total weighted flow time of all n jobs, where there is a weight associated with each job. We establish the necessary and sufficient condition for the existence of a job permutation which remains optimal for any possible realizations of these n uncertain processing times. We also establish the necessary and sufficient condition for another extreme case that for each of the $n!$ job permutations there exists a possible realization of the uncertain processing times that this permutation is uniquely optimal. Testing each of the conditions takes polynomial time in terms of the number n of jobs. We develop precedence–dominance relations among the n jobs in dealing with the general case of this uncertain scheduling problem. In case there exist no precedence–dominance relations among a subset of n jobs, a heuristic procedure to minimize the maximal absolute or relative regret is used for sequencing such a job subset. Computational experiments for randomly generated instances with n jobs $(5 \leq n \leq 1000)$ show that the established precedence–dominance relations are quite useful in reducing the total weighted flow time.

1. Introduction

Most real-life scheduling problems involve some forms of uncertainty. In dealing with an uncertain parameter (say, an uncertain processing time) of a scheduling problem, a traditional approach is to assume that the uncertain parameter is a random variable with a specific probability distribution (see the second part of monograph [1]). It is worthwhile to mention that in many real-life scheduling situations with an uncertain parameter, we may have no sufficient information to characterize the probability distribution for the uncertain parameter. Therefore, different approaches and techniques are needed [2]. A variational inequality approach [3] was proposed with an application to the calculation of dynamic traffic network. A stability radius with a two-stage (off-line planning and on-line scheduling) scheduling approach [4,5] was proposed. A robust schedule was proposed [6–11] for a decision-maker who prefers a solution that hedges against the worst possible scenario.

Two types of robustness have been distinguished for a scheduling problem with uncertain processing times: Quality robustness and solution robustness [6]. The former is a property of the schedule whose objective function value does not deviate much from optimality while small changes in the job processing times occur. The latter can be described as robustness in the solution space. A decision-maker might be forced to modify the original schedule when changes in the

* Corresponding author.
E-mail address: sotskov2005@hotmail.com (Yu.N. Sotskov).

0895-7177/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mcm.2009.03.006

job processing times occur. Thus, quality robustness is a property of a schedule that is more or less stable with respect to changes of the job processing times before these changes occur (off-line scheduling), while solution robustness refers to the schedule stability after changes of the job processing times have occurred (on-line scheduling).

For both types of robustness, it is useful to construct a minimal set of dominant schedules which optimally cover all possible scenarios (possible realizations of the uncertain job processing times) in the sense that for any possible scenario there exists at least one schedule in this dominant set which is optimal. If this dominant set consists of one schedule (the simplest case of off-line scheduling), then this schedule will remain optimal for any possible scenario. In the general case, the set of dominance schedules enables a decision-maker to quickly make an on-line scheduling decision whenever additional local information on a realization of an uncertain job processing time becomes available.

In this paper, we consider a single-machine scheduling problem of n jobs in which each uncertain job processing time is independent and can take any value within each corresponding (closed) interval. There is a weight associated with each job. The scheduling objective is to minimize the weighted sum of the job completion times.

The paper is organized as follows. In Section 2, the problem settings and a literature review are given. Section 3 contains some known results for a single-machine scheduling problem with fixed processing times. Section 4 provides a necessary and sufficient condition in which one job can precede another in an optimal schedule for any possible realization of the uncertain job processing times. Section 5 contains a necessary and sufficient condition for an extreme case in which there exists a job permutation remaining optimal for all possible realizations of the uncertain processing times. Section 5 also contains a necessary and sufficient condition for the other extreme case in which there exists a possible realization of the uncertain processing times for each of the $n!$ job permutations to be uniquely optimal. Illustrative examples are discussed in Section 6. Section 7 provides algorithms for robust scheduling, i.e., worst-case absolute or relative deviation from optimality is minimized. Computational results for randomly generated instances of a single-machine scheduling problem with uncertain (interval) processing times are given in Section 8. We conclude with Section 9.

2. Problem settings and state-of-the-art

There are $n \geq 2$ jobs $\mathcal{J} = \{J_1, J_2, \dots, J_n\}$ to be processed on a single machine. For each job $J_i \in \mathcal{J}$, a positive weight $w_i > 0$ is given. The processing time p_i of job $J_i \in \mathcal{J}$ is unknown before scheduling and may take any real value between a given lower bound $p_i^L > 0$ and an upper bound $p_i^U \geq p_i^L$. In other words, the set T of all possible vectors $p = (p_1, p_2, \dots, p_n)$ of the uncertain processing times is a rectangular box in the space R^n_+ of non-negative n-dimensional real vectors:

$$T = \{p \mid p \in R^n_+, p_i^L \leq p_i \leq p_i^U\}. \tag{1}$$

Let $C_i = C_i(\pi_k, p)$ denote the completion time of job $J_i \in \mathcal{J}$ in the semi-active schedule defined by permutation π_k of n jobs from set $\mathcal{J}$, provided that vector $p \in T$ of job processing times is fixed. Criterion $\sum w_iC_i$ denotes a minimization of the sum of the weighted completion times:

$$\sum_{J_i \in \mathcal{J}} w_iC_i(\pi_t, p) = \min_{\pi_k \in S} \left\{ \sum_{J_i \in \mathcal{J}} w_iC_i(\pi_k, p) \right\},$$

where $S = \{\pi_1, \pi_2, \dots, \pi_{n!}\}$ is a set of all permutations $\pi_k = (J_{k_1}, J_{k_2}, \dots, J_{k_n})$ of n jobs from set $\mathcal{J} = \{J_1, J_2, \dots, J_n\} = \{J_{k_1}, J_{k_2}, \dots, J_{k_n}\}$, and permutation $\pi_t \in S$ is optimal. By adopting the three-field notation $\alpha|\beta|\gamma$ introduced in [12], we denote the problem of finding a permutation π_t that minimizes the value $\sum_{J_i \in \mathcal{J}} w_iC_i(\pi_k, p)$ among permutations $\pi_k \in S$ as $1|p_i^L \leq p_i \leq p_i^U| \sum w_iC_i$. Since vector $p \in T$ of the processing times is unknown before scheduling, the completion time C_i of the job $J_i \in \mathcal{J}$ cannot be calculated before scheduling. Mathematically, the above problem $1|p_i^L \leq p_i \leq p_i^U| \sum w_iC_i$ is not correct. In OR literature, different approaches for correcting optimization problems like $1|p_i^L \leq p_i \leq p_i^U| \sum w_iC_i$ were proposed. Next, we survey some settings of the scheduling problems with uncertain processing times.

If a vector $p \in T$ of the processing times is fixed before scheduling (i.e., equality $p_i^L = p_i^U$ holds for each job $J_i \in \mathcal{J}$), then problem $1|p_i^L \leq p_i \leq p_i^U| \sum w_iC_i$ reduces to the conventional problem $1 \parallel \sum w_iC_i$ with the fixed processing times. The latter problem $1 \parallel \sum w_iC_i$ is mathematically correct and can be solved in $O(n \log_2 n)$ time [13].

In what follows, we will call the problem $\alpha|p_i^L \leq p_i \leq p_i^U|\gamma$ with objective function $\gamma = f(C_1, C_2, \dots, C_n)$ an *uncertain* problem in contrast to its *deterministic* counterpart, problem $\alpha \parallel \gamma$. We note that for the uncertain problem $\alpha|p_i^L \leq p_i \leq p_i^U|\gamma$, there may not exist a single permutation of n jobs that remains optimal for all possible realizations of the job processing times. Therefore, it is necessary to introduce an additional criterion in dealing with the uncertain problem $\alpha|p_i^L \leq p_i \leq p_i^U|\gamma$. In [8,9], a *robust schedule* minimizing the worst-case absolute (relative) deviation from optimality was proposed to hedge against processing time uncertainty. In [8,10,11], the uncertain problem $1|p_i^L \leq p_i \leq p_i^U| \sum C_i$ of minimizing the total flow time (i.e., $w_i = 1$ for each job $J_i \in \mathcal{J}$) has been considered. In [7,8,10,11,14–16], in contrast with the *continuous* intervals of the processing times defined by (1), the processing time uncertainty was described through a *finite discrete* set T^*, $|T^*| = m$, of possible processing time vectors (scenarios). Each scenario $p^j = (p_1^j, p_2^j, \dots, p_n^j) \in T^*, j \in \{1, 2, \dots, m\}$, represents a set of fixed processing times for jobs from set $\mathcal{J}$, which can be realized with some positive (but unknown before scheduling) probability. For a specific scenario $p^j \in T^*$, the deterministic problem $1 \parallel \sum C_i$ arises which can be solved using the SPT rule [13]: Sort the jobs from set $\mathcal{J}$ in non-decreasing order of the processing times $p_i^j, J_i \in \mathcal{J}$.

Let $\gamma_{p^j}^t$ (with $p^j \in T^*$) denote the optimal value of the objective function $\gamma = f(C_1, C_2, \dots, C_n)$ for the deterministic problem $1 \parallel \gamma$ with vector p^j of the job processing times. Let permutation $\pi_t \in S$ be optimal:

$$f(C_1(\pi_t, p^j), C_2(\pi_t, p^j), \dots, C_n(\pi_t, p^j)) = \gamma_{p^j}^t = \min_{\pi_k \in S} \gamma_{p^j}^k = \min_{\pi_k \in S} f(C_1(\pi_k, p^j), C_2(\pi_k, p^j), \dots, C_n(\pi_k, p^j)).$$

For any permutation $\pi_k \in S$ and any scenario $p^j \in T^*$, the difference $\gamma_{p^j}^k - \gamma_{p^j}^t = r(\pi_k, p^j)$ is called the *regret* for permutation π_k with objective function value equal to $\gamma_{p^j}^k$ under scenario p^j. For any permutation $\pi_k \in S$, value

$$Z(\pi_k) = \max\{r(\pi_k, p^j) \mid p^j \in T^*\}$$

is called a *worst-case absolute regret*. A *worst-case relative regret* is defined as follows:

$$Z'(\pi_k) = \max\left\{\frac{r(\pi_k, p^j)}{\gamma_{p^j}^t} \mid p^j \in T^*\right\},$$

where $\gamma_{p^j}^t \neq 0$. While the deterministic problem $1 \parallel \sum C_i$ is computationally simple [13], finding a job permutation of minimizing the worst-case regret to the uncertain counterpart with discrete set T^* of possible scenarios is computationally hard. In [8], finding a permutation $\pi_k \in S$ of minimizing the worst-case absolute regret $Z(\pi_k)$ was shown to be binary NP-hard even for two possible scenarios: $|T^*| = 2$. In [11], it was shown that finding a permutation $\pi_k \in S$ of minimizing the worst-case relative regret $Z'(\pi_k)$ is binary NP-hard even for two possible scenarios (binary NP-complete two-partition problem has been polynomially reduced to recognition version of minimizing $Z'(\pi_k)$), while the problem is unary NP-hard for an unbounded number m of possible scenarios (unary NP-complete three-partition problem has been polynomially reduced to recognition version of minimizing $Z(\pi_k)$ or $Z'(\pi_k)$).

Similarly, worst-case regrets can be defined for a rectangular box $T \in R_+^n$ of vectors $p \in T$ of the possible processing times. We save the same notations $Z(\pi_k)$ and $Z'(\pi_k)$ for worst-case absolute and relative regrets, respectively, if n (closed) intervals of possible processing times are assumed to be given:

$$Z(\pi_k) = \max\{r(\pi_k, p) \mid p \in T\}; \tag{2}$$

$$Z'(\pi_k) = \max\left\{\frac{r(\pi_k, p)}{\gamma_{p^j}^t} \mid p \in T\right\}, \quad \gamma_{p^j}^t \neq 0. \tag{3}$$

In [10], it was proven that minimizing the worst-case absolute regret $Z(\pi_k)$ for problem $1|p_i^L \le p_i \le p_i^U| \sum C_i$ is binary NP-hard even if the intervals $[p_i^L, p_i^U]$ of the possible processing times have the same center in the real axis for all jobs $J_i \in \mathcal{J}$. It should be noted that in [14], it was shown (by example) that there is no direct relationship between a given finite discrete set T^* and infinite continuous set T in the aspect of the complexity of the uncertain problem.

In [17], binary NP-hardness was proven for finding a permutation $\pi_k \in S$ that minimizes the worst-case absolute regret $Z(\pi_k)$ for the uncertain two-machine flow-shop problem with the criterion $C_{\max}$ of minimum makespan:

$$\max\{C_{\max}(\pi_t, p) \mid J_i \in \mathcal{J}\} = \min_{\pi_k \in S}\{\max\{C_i(\pi_k, p) \mid J_i \in \mathcal{J}\}\},$$

even for two possible scenarios: $|T^*| = 2$.

There are a few polynomially solvable cases for the uncertain scheduling problems. In [18], an algorithm with complexity $O(n^4)$ has been developed for minimizing the worst-case regret for the problem $1|p_i^L \le p_i \le p_i^U, d_i^L \le d_i \le d_i^U|L_{\max}$ with the criterion $L_{\max}$ of minimizing the maximum lateness:

$$\max\{C_i(\pi_t, p) - d_i \mid J_i \in \mathcal{J}\} = \min_{\pi_k \in S}\{\max\{C_i(\pi_k, p) - d_i \mid J_i \in \mathcal{J}\}\},$$

when both intervals of the possible processing times p_i and the possible due dates d_i are given as input data. In [10], it was proven that minimizing $Z(\pi_k)$ for the problem $1|p_i^L \le p_i \le p_i^U| \sum C_i$ can be realized in $O(n \log_2 n)$ time, if all the given intervals $[p_i^L, p_i^U], J_i \in \mathcal{J}$, of the possible processing times have the same center in the real axis and the number n of jobs is even.

In summary, it is observed that for the most classical polynomially solvable deterministic scheduling problems, their uncertain counterparts with minimizing the worst-case regret become binary or unary NP-hard. Indeed, even for the simple case of only two possible scenarios of the job processing times, minimizing $Z(\pi_k)$ or $Z'(\pi_k)$ implies a time-consuming search over the set S of $n!$ permutations of n jobs. In order to overcome this computational complexity in some cases, we combine the worst-case regret concept with the concept of the *minimal set of dominant schedules* introduced in [4,5] for solving uncertain job-shop problem $J|p_i^L \le p_i \le p_i^U|C_{\max}$.

Definition 1. A set of permutations (semi-active schedules) $S(T) \subseteq S$ is a *minimal dominant set* for the uncertain problem $\alpha|p_i^L \leq p_i \leq p_i^U|\gamma$, if for any fixed vector $p \in T$ set $S(T)$ contains at least one permutation (semi-active schedule), which is optimal for the deterministic problem $\alpha \parallel \gamma$ associated with this vector p of the job processing times, provided that any proper subset of set $S(T)$ loses such a property.

A minimal dominant set $S(T)$ was investigated in [5,19–21] for the $C_{\max}$ criterion, and in [20,22] for the total flow time criterion $\sum C_i$. In particular, article [22] addresses the total flow time criterion in a two-machine flow-shop problem $F2|p_i^L \leq p_i \leq p_i^U|\sum C_i$. A geometrical algorithm has been developed for solving the flow-shop problem $Fm|p_i^L \leq p_i \leq p_i^U, n = 2|\sum C_i$ with m machines and two jobs. For an uncertain flow-shop problem with two or three machines, sufficient conditions have been identified when the transposition of two jobs minimizes the total flow time. The work of [20] deals with the case of separate setup times with the criterion of minimizing the makespan or the total flow time. Namely, the processing times were fixed while each setup time was relaxed to be a distribution-free random variable within a given lower and upper bound. Dominance relations have been identified for an uncertain flow-shop problem with two machines. In [19], for a two-machine flow-shop problem $F2|p_i^L \leq p_i \leq p_i^U|C_{\max}$ sufficient conditions have been identified when the transposition of two jobs minimizes the makespan. In [21], the necessary and sufficient conditions were proven for the case when a single semi-active schedule dominates all the others, and the necessary and sufficient conditions were proven for the case when it is possible to fix the optimal order of two jobs for the makespan criterion with interval processing times.

In contrast to references [4,5], where the exponential algorithms based on an exhaustive enumeration of all semi-active schedules were derived for constructing a minimal dominant set $S(T)$ for the uncertain shop scheduling problems, in Sections 4 and 5, we show how to obtain a minimal dominant set $S(T)$ for the uncertain single-machine problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$ in polynomial time. In the next section, Section 3, we first present the auxiliary results for the deterministic problem $1 \parallel \sum w_iC_i$.

3. Fixed processing times

In [13], it was proven that the deterministic problem $1 \parallel \sum w_iC_i$ can be solved in $O(n \log_2 n)$ time due to the following sufficient condition for the optimality of permutation $\pi_k = (J_{k_1}, J_{k_2}, \ldots, J_{k_n}) \in S$:

$$\frac{w_{k_1}}{p_{k_1}} \geq \frac{w_{k_2}}{p_{k_2}} \geq \cdots \geq \frac{w_{k_n}}{p_{k_n}}. \tag{4}$$

Hereafter, inequality $p_{k_i} > 0$ must hold for each job $J_{k_i} \in \mathcal{J}$. Note that inequalities (4) are also a necessary condition for the optimality of permutation $\pi_k \in S$ as follows from the following theorem which can be found in [23].

Theorem 1. *Permutation $\pi_k = (J_{k_1}, J_{k_2}, \ldots, J_{k_n}) \in S$ is optimal for the problem $1 \parallel \sum w_iC_i$ if and only if inequalities (4) hold.*

To obtain the corresponding result for the uncertain problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$ we need the following two corollaries from Theorem 1.

Corollary 1. *If inequality $\frac{w_u}{p_u} > \frac{w_v}{p_v}$ holds, then in all optimal permutations for the problem $1 \parallel \sum w_iC_i$ job J_u precedes job J_v.*

Proof. By contradiction, we assume that there exists such an optimal permutation $\pi_m \in S$ for the problem $1 \parallel \sum w_iC_i$ that inequality $\frac{w_u}{p_u} > \frac{w_v}{p_v}$ holds for jobs $J_u \in \mathcal{J}$ and $J_v \in \mathcal{J}$. However, job J_u follows after job J_v in permutation π_m.

Necessity of condition (4) in Theorem 1 implies inequalities $\frac{w_v}{p_v} \geq \frac{w_{v+1}}{p_{v+1}} \geq \cdots \geq \frac{w_u}{p_u}$, where it is assumed that $v < u$. Thus, we obtain inequality $\frac{w_v}{p_v} \geq \frac{w_u}{p_u}$ which contradicts the above condition $\frac{w_u}{p_u} > \frac{w_v}{p_v}$. Corollary 1 is proven. ■

Corollary 2. *If equality $\frac{w_u}{p_u} = \frac{w_v}{p_v}$ holds, then the problem $1 \parallel \sum w_iC_i$ has both optimal permutation $\pi_l \in S$ of the form*

$$\pi_l = (\ldots, J_u, J_v, \ldots) \tag{5}$$

and optimal permutation $\pi_m \in S$ of the form

$$\pi_m = (\ldots, J_v, J_u, \ldots). \tag{6}$$

Proof. Since the set S is finite, there exists at least one permutation π_l of the form either

$$\pi_l = (\ldots, J_u, \ldots, J_v, \ldots) \quad \text{or} \quad \pi_l = (\ldots, J_u, J_v, \ldots),$$

or permutation π_m of the form either

$$\pi_m = (\ldots, J_v, \ldots, J_u, \ldots) \quad \text{or} \quad \pi_m = (\ldots, J_v, J_u, \ldots)$$

which is optimal for the problem $1 \parallel \sum w_iC_i$ under consideration. Since equality $\frac{w_u}{p_u} = \frac{w_v}{p_v}$ holds, a part of the necessary and sufficient condition (4) of optimality of permutation π_l looks as follows:

$$\cdots \geq \frac{w_u}{p_u} = \cdots = \frac{w_v}{p_v} \geq \ldots, \tag{7}$$

and a part of the necessary and sufficient condition (4) of optimality of permutation π_m looks as follows:

$$\cdots \geq \frac{w_v}{p_v} = \cdots = \frac{w_u}{p_u} \geq \cdots. \tag{8}$$

Obviously, if condition (7) holds, then condition (8) holds without fail, and vice versa. Hence, for the problem $1 \parallel \sum w_iC_i$ under consideration, there exist both optimal permutation of the form either

$$\pi_l = (\ldots, J_u, \ldots, J_v, \ldots) \quad \text{or} \quad \pi_l = (\ldots, J_u, J_v, \ldots),$$

and optimal permutation of the form either

$$\pi_m = (\ldots, J_v, \ldots, J_u, \ldots) \quad \text{or} \quad \pi_m = (\ldots, J_v, J_u, \ldots).$$

To complete the proof we have to show that if permutation of the form $\pi_l = (\ldots, J_u, \ldots, J_v, \ldots)$ is optimal, then there also exists an optimal permutation of the form $(\ldots, J_u, J_v, \ldots)$. Indeed, if permutation $\pi_l = (\ldots, J_u, \ldots, J_v, \ldots)$ is optimal, then due to Theorem 1 for any job J_k locating between jobs J_u and J_v in permutation π_l the following equalities must hold: $\frac{w_u}{p_u} = \frac{w_k}{p_k} = \frac{w_v}{p_v}$. Therefore, permutation obtained from permutation π_l via rearranging jobs J_k and J_v will be also optimal. After a finite number of such rearrangements, we obtain an optimal permutation of the form $(\ldots, J_u, J_v, \ldots)$. This completes the proof. ■

The above results valid for the deterministic problem $1 \parallel \sum w_iC_i$ will be used for finding a minimal dominant set $S(T)$ for the uncertain problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$ in general case (Section 4) and in the two special cases, i.e., when $|S(T)| = 1$ or $|S(T)| = n!$ (Section 5).

4. Uncertain processing times

A minimal dominant set $S(T)$ for uncertain problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$ will be obtained by constructing a precedence–dominance relation on the set of jobs $\mathcal{J}$. We define a precedence–dominance relation as follows.

Definition 2. Job J_u dominates job J_v with respect to T (denoted by $J_u \mapsto J_v$) if there exists a minimal dominant set $S(T)$ for the problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$ such that job J_u precedes job J_v in every permutation of set $S(T)$.

From Definition 2, it follows that a minimal dominant set $S(T)$ constructed for the deterministic problem $1 \parallel \sum w_iC_i$ associated with the vector $p \in T$ of the job processing times is a singleton: $S(T) = \{\pi_k\}$, where set T is also a singleton, $T = \{p\}$. As a result precedence–dominance relations $J_{k_1} \mapsto J_{k_2} \mapsto J_{k_3} \mapsto \cdots \mapsto J_{k_{n-1}} \mapsto J_{k_n}$ with respect to set $T = \{p\}$ hold for the deterministic problem $1 \parallel \sum w_iC_i$.

For an uncertain problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$, we prove the following claim.

Theorem 2. *For the problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$, job J_u dominates job J_v with respect to T if and only if the following inequality holds:*

$$\frac{w_u}{p_u^U} \geq \frac{w_v}{p_v^L}. \tag{9}$$

Proof. *Sufficiency.* Let inequality (9) hold. Then we need to prove that job J_u dominates job J_v with respect to T, i.e., there exists a minimal dominant set $S(T)$ for the problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$ such that job J_u precedes job J_v in every permutation $\pi_k \in S(T)$.

Since the set S is finite, we can construct a minimal dominant set $S'(T) \subseteq S$ for the problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$ via consecutively deleting redundant permutations from set S. Let us take any vector $p \in T$ of the job processing times. Inequalities $p_u^L \leq p_u \leq p_u^U$ imply

$$\frac{w_u}{p_u^L} \geq \frac{w_u}{p_u} \geq \frac{w_u}{p_u^U}. \tag{10}$$

Inequalities $p_v^L \leq p_v \leq p_v^U$ imply

$$\frac{w_v}{p_v^L} \geq \frac{w_v}{p_v} \geq \frac{w_v}{p_v^U}. \tag{11}$$

From inequalities (9)–(11) we obtain inequalities

$$\frac{w_u}{p_u} \geq \frac{w_u}{p_u^U} \geq \frac{w_v}{p_v^L} \geq \frac{w_v}{p_v},$$

i.e., for any possible processing times $p_u \in [p_u^L, p_u^U]$ and $p_v \in [p_v^L, p_v^U]$, it follows:

$$\frac{w_u}{p_u} \geq \frac{w_v}{p_v}. \tag{12}$$

If at least one of the three strict inequalities $p_u < p_u^U$, $p_v^L < p_v$, or $\frac{w_u}{p_u} > \frac{w_v}{p_v}$ holds, then (12) also becomes a strict inequality: $\frac{w_u}{p_u} > \frac{w_v}{p_v}$. Then due to Corollary 1 job J_u precedes job J_v in any optimal permutation $\pi_k \in S$ for the deterministic problem $1 \parallel \sum w_iC_i$ associated with the vector $p \in T$ of the job processing times. Hence set $S'(T)$ has to include permutation $\pi_k \in S$ of the form either $\pi_k = (\ldots, J_u, \ldots, J_v, \ldots)$ or $\pi_k = (\ldots, J_u, J_v, \ldots)$.

If for a vector $p = (p_1, p_2, \ldots, p_n)$ all three equalities $p_u = p_u^U$, $p_v^L = p_v$, and $\frac{w_u}{p_u} = \frac{w_v}{p_v}$ hold, then due to Corollary 2 there exist both optimal permutation $\pi_l \in S$ of the form $\pi_l = (\ldots, J_u, J_v, \ldots)$ and optimal permutation $\pi_m \in S$ of the form $\pi_m = (\ldots, J_v, J_u, \ldots)$ for the deterministic problem $1 \parallel \sum w_iC_i$ associated with the vector $p \in T$ of the job processing times. If for all such vectors $p \in T$ set $S'(T)$ contains permutation $\pi_l \in S$ of the form (5): $\pi_l = (\ldots, J_u, J_v, \ldots)$, then set $S'(T)$ is a desired set: $S'(T) = S(T)$. Otherwise if for some vector $p \in T$ set $S'(T)$ contains permutation $\pi_m \in S$ of the form (6): $\pi_m = (\ldots, J_v, J_u, \ldots)$, then we replace the permutation of the form (6) by the permutation of the form (5). Due to Corollary 1, the later permutation is also an optimal one for the deterministic problem $1 \parallel \sum w_iC_i$ associated with the vector $p \in T$ of the job processing times. Having done such replacement of the permutation $\pi_m \in S$ by the permutation $\pi_l \in S$ for all specified equalities $p_u = p_u^U$, $p_v^L = p_v$, and $\frac{w_u}{p_u} = \frac{w_v}{p_v}$, we transform the set $S'(T)$ to the desired minimal dominant set $S(T)$ for the problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$. Thus, job J_u precedes job J_v in each permutation $\pi_k \in S(T)$. In other words, job J_u dominates job J_v with respect to T. Sufficiency of condition (9) is proven.

Necessity. We prove necessity of condition (9) by contradiction. Let job J_u dominate job J_v with respect to T, i.e., let there exist a minimal dominant set $S(T)$ for the problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$ such that job J_u precedes job J_v in each permutation $\pi_k \in S(T)$. However, let condition (9) be violated, i.e., let opposite inequality $\frac{w_u}{p_u^U} < \frac{w_v}{p_v^L}$ hold. Let us show that there exist such possible processing times p'_u (with $p_u^L \leq p'_u \leq p_u^U$) and p'_v (with $p_v^L \leq p'_v \leq p_v^U$) for which inequality $\frac{w_u}{p'_u} < \frac{w_v}{p'_v}$ holds. Indeed, we can set $p'_u = p_u^U$ and $p'_v = p_v^L$ and obtain

$$\frac{w_u}{p'_u} = \frac{w_u}{p_u^U} < \frac{w_v}{p_v^L} = \frac{w_v}{p'_v}.$$

Thus, for these processing times p'_u and p'_v, inequality $\frac{w_u}{p'_u} < \frac{w_v}{p'_v}$ holds. Due to Corollary 1, there is no optimal permutation $\pi_l \in S(T)$ to the deterministic problem $1 \parallel \sum w_iC_i$ with such processing times p'_u and p'_v such that job J_u precedes job J_v in permutation π_l. Hence, the above set $S(T)$ cannot be a minimal dominant set for the uncertain problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$ under consideration (since this set $S(T)$ does not contain an optimal permutation for the possible vector of the job processing times with components $p'_u = p_u^U$ and $p'_v = p_v^L$). Theorem 2 is proven. ■

Theorem 2 allows us to find a minimal dominant set $S(T)$ for the problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$ in a compact manner. To this end, we check condition (9) for each pair of jobs J_u and J_v from set $\mathcal{J}$ and construct a digraph $(\mathcal{J}, \mathcal{A})$ of precedence–dominance relation on set $\mathcal{J}$ as follows: Arc (J_u, J_v) belongs to set $\mathcal{A}$ if and only if dominance relation $J_u \mapsto J_v$ holds. Set of arcs $\mathcal{A} \subseteq \mathcal{J} \times \mathcal{J}$ of digraph $(\mathcal{J}, \mathcal{A})$ defines the transitive binary relation on set $\mathcal{J}$. To construct a digraph $(\mathcal{J}, \mathcal{A})$ takes $O(n^2)$ time. If for all jobs $J_i \in \mathcal{J}$ inequality $p_i^L < p_i^U$ holds, then the digraph $(\mathcal{J}, \mathcal{A})$ defines the strict order relation on set $\mathcal{J}$.

On the other hand, if there are two jobs $J_u \in \mathcal{J}$ and $J_v \in \mathcal{J}$ (or more than two such jobs) for which their given intervals of the possible processing times degenerate into a point:

$$\frac{w_u}{p_u^L} = \frac{w_u}{p_u^U} = \frac{w_v}{p_v^L} = \frac{w_v}{p_v^U}, \tag{13}$$

then contour (contours) arises in the digraph $(\mathcal{J}, \mathcal{A})$. It is clear that the order of such jobs J_u and J_v has no influence on the value of the objective function $\gamma = \sum_{i=1}^{n} w_iC_i$. Therefore one can fix an order of these two jobs. For example, to exclude contours from digraph $(\mathcal{J}, \mathcal{A})$ we accept the following agreement: If equalities (13) hold and $u < v$, then arc (J_v, J_u) is deleted from the set of arcs $\mathcal{A}$. Having done such deletions of arcs for all specified equalities (13), we obtain the strict order relation $\mathcal{A}^* \subset \mathcal{A} \subseteq \mathcal{J} \times \mathcal{J}$ on the set of jobs $\mathcal{J}$. Let digraph $(\mathcal{J}, \mathcal{A}^*)$ be denoted as G: $G = (\mathcal{J}, \mathcal{A}^*)$. If digraph $(\mathcal{J}, \mathcal{A})$ has no contour, we denote it as G, too. Note that instead of using digraph G, it is often useful to adopt a reduction $G^0 = (\mathcal{J}, \mathcal{A}^0)$ of digraph G. Digraph G^0 is obtained from G via deleting transitive arcs.

5. Extreme cases of an uncertain problem

The cardinality of the minimal dominant set $S(T)$ may be considered as a measure of the uncertainty of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$: The uncertainty in the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ is less, if the cardinality of set $S(T)$ is smaller.

Inclusion $S(T) \subseteq S$ implies inequalities $1 \le |S(T)| \le n!$. In this section, we present the characterizations of two extreme cases of a minimal dominant set.

5.1. *Dominant singleton:* $|S(T)| = 1$

The simplest case of an uncertain problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ arises, when equality $|S(T)| = 1$ holds. In such a case, a minimal dominant set for the uncertain problem is a singleton: $\{\pi_k\} = S(T)$ (as well as an ordinary solution to a deterministic problem $1 \parallel \sum w_iC_i$).

Theorem 3. *For an existence of a dominant singleton $S(T) = \{\pi_k\} = \{(J_{k_1}, J_{k_2}, \dots, J_{k_n})\}$ for the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$, it is necessary and sufficient that the following set of inequalities holds:*

$$\frac{w_{k_1}}{p_{k_1}^U} \ge \frac{w_{k_2}}{p_{k_2}^L}, \frac{w_{k_2}}{p_{k_2}^U} \ge \frac{w_{k_3}}{p_{k_3}^L}, \dots, \frac{w_{k_{n-1}}}{p_{k_{n-1}}^U} \ge \frac{w_{k_n}}{p_{k_n}^L}. \tag{14}$$

Proof. *Sufficiency.* Let condition (14) hold. In this case, sufficiency of condition (9) in Theorem 2 implies the following precedence–dominance relation: $J_{k_1} \mapsto J_{k_2} \mapsto J_{k_3} \mapsto \cdots \mapsto J_{k_{n-1}} \mapsto J_{k_n}$ with respect to T. Hence, according to Definition 2, there exists a minimal dominant set $S(T)$ for the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ in which all permutations look as follows: $(J_{k_1}, J_{k_2}, J_{k_3}, \dots, J_{k_{n-1}}, J_{k_n})$. As a result, we obtain a singleton: $S(T) = \{\pi_k\}$.

Necessity. Let a singleton $\{\pi_k\}$ be a minimal dominant set $S(T)$ for the uncertain problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$, i.e., for any fixed vector $p \in T$ permutation π_k is optimal for the deterministic problem $1 \parallel \sum w_iC_i$ associated with the vector p of the job processing times. Let us show that inequalities (14) should hold.

First, we show that the first inequality from (14) should hold. For this purpose, as a vector $p \in T$, we take a vector with the first and second components defined as follows: $p_1 := p_{k_1}^U$ and $p_2 := p_{k_2}^L$. Other components p_i, $i \in \{3, 4, \dots, n\}$, of the vector p can be arbitrary: $p_i^L \le p_i \le p_i^U$. Since permutation π_k is optimal for the deterministic problem $1 \parallel \sum w_iC_i$ associated with the vector p of the job processing times, the necessity of condition (4) in Theorem 1 implies the first inequality in (14):

$$\frac{w_{k_1}}{p_{k_1}^U} \ge \frac{w_{k_2}}{p_{k_2}^L}.$$

After $(n-1)$ steps, we can obtain other inequalities from condition (14):

$$\frac{w_{k_j}}{p_{k_j}^U} \ge \frac{w_{k_{j+1}}}{p_{k_{j+1}}^L},$$

via fixing consecutive components $p_j := p_{k_j}^U$ and $p_{j+1} := p_{k_{j+1}}^L$, $j \in \{2, 3, \dots, n-1\}$, in the vector $p \in T$. Theorem 3 is proven. ∎

Obviously, if a dominant singleton $S(T) = \{\pi_k\}$ for the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ exists, then a dominant singleton can be obtained using Theorem 2 in $O(n^2)$ time. Next, we show that as an application for this purpose Theorem 3 allows us either to find a singleton $S(T)$ (if it exists) faster or to prove that a dominant singleton does not exist.

For a fast checking of condition (14) of Theorem 3, we sort jobs of set $\mathcal{J}$ in non-increasing order of the fractions $\frac{w_{k_j}}{\overline{p}_{k_j}}$, where $\overline{p}_{k_j}$ denotes the midpoint of the segment $[p_{k_j}^L, p_{k_j}^U]$:

$$\overline{p}_{k_j} = \frac{p_{k_j}^U - p_{k_j}^L}{2}.$$

As a result, we obtain a permutation $\pi_k = (J_{k_1}, J_{k_2}, \dots, J_{k_n}) \in S$ in $O(n \log_2 n)$ time. Due to sufficiency in Theorem 3, if condition (14) holds, then this permutation π_k defines a dominant singleton $S(T) = \{\pi_k\}$ for the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$.

On the other hand, it is easy to show that if conditions (14) does not hold for all pairs of consecutive jobs in permutation $\pi_k = (J_{k_1}, J_{k_2}, \dots, J_{k_n})$, then a dominant singleton for the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ does not exist. It takes $O(n)$ time to check conditions (14) for a fixed permutation π_k of n jobs. Thus, the asymptotic complexity of checking condition of Theorem 3 is estimated by $O(n \log_2 n)$.

5.2. Minimal dominant set with the maximal cardinality: $|S(T)| = n!$

The most uncertain case (in the sense of Definition 1) of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ arises when $|S(T)| = n!$.

Theorem 4. *Let $p_i^L < p_i^U, J_i \in \mathcal{J}$. For an existence of a minimal dominant set $S(T)$ for the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ with the maximal cardinality $|S(T)| = n!$, it is necessary and sufficient that the following inequality holds:*

$$\max\left\{\frac{w_i}{p_i^U} \mid J_i \in \mathcal{J}\right\} < \min\left\{\frac{w_i}{p_i^L} \mid J_i \in \mathcal{J}\right\}. \tag{15}$$

Proof. *Sufficiency.* We denote $a = \max\{\frac{w_i}{p_i^U} \mid J_i \in \mathcal{J}\}$ and $b = \min\{\frac{w_i}{p_i^L} \mid J_i \in \mathcal{J}\}$. Let inequality (15) hold: $a < b$. We choose any permutation $\pi_k = (J_{k_1}, J_{k_2}, \ldots, J_{k_n}) \in S$ of n jobs and show that this permutation has to belong to any minimal dominant set $S(T)$ constructed for the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ under consideration.

To this end, we show that there exists a vector $p^* = (p_1^*, p_2^*, \ldots, p_n^*) \in T$ such that the following inequalities hold:

$$\frac{w_{k_1}}{p_{k_1}^*} > \frac{w_{k_2}}{p_{k_2}^*} > \cdots > \frac{w_{k_n}}{p_{k_n}^*}. \tag{16}$$

Indeed, due to the strict inequality (15), the length of the segment $[a, b]$, $a < b$, is strictly positive, and moreover, the segment $[a, b]$ is equal to the intersection of n segments

$$\left[\frac{w_i}{p_i^U}, \frac{w_i}{p_i^L}\right], i \in \{1, 2, \ldots, n\}, \quad \text{namely: } [a, b] = \bigcap_{i=1}^{n}\left[\frac{w_i}{p_i^U}, \frac{w_i}{p_i^L}\right].$$

Therefore, since segment $[a, b]$, $a < b$, of non-negative real numbers is dense everywhere, it is possible to find n real numbers $p_{k_i}^*, J_{k_i} \in \mathcal{J}$, which satisfy all inequalities (16). Thus, due to Corollary 1, in any optimal permutation for the deterministic problem $1 \parallel \sum w_iC_i$ associated with the vector p^* of the job processing times, job J_{k_1} precedes job J_{k_2}, job J_{k_2} precedes job J_{k_3}, and so on, job $J_{k_{n-1}}$ precedes job J_{k_n}. Therefore, permutation π_k is the unique optimal permutation for the problem $1 \parallel \sum w_iC_i$ associated with the vector p^* of the job processing times. Hence, according to Definition 1 any minimal dominant set $S(T)$ constructed for the uncertain problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ necessarily contains permutation π_k. Since permutation π_k has been chosen arbitrarily in set S, any minimal dominant set $S(T)$ contains all permutations of set S, i.e., set $S(T)$ coincides with set S. As a result we obtain $|S(T)| = |S| = n!$.

Necessity. Let $|S(T)| = n!$. Hence $|S(T)| = |S|$. We have to show that inequality (15) holds. By contradiction, we assume that inequality (15) does not hold.

Hence, there exist at least two jobs $J_u \in \mathcal{J}$ and $J_v \in \mathcal{J}$ such that inequality

$$\frac{w_u}{p_u^U} \ge \frac{w_v}{p_v^L}$$

holds. Then from the sufficiency of condition (9) in Theorem 2, it follows that jobs J_u dominates jobs J_v with respect to T, i.e., there exists a minimal dominant set $S'(T)$ for the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ such that all permutations in $S'(T)$ look like $(\ldots, J_u, \ldots, J_v, \ldots)$ or $(\ldots, J_u, J_v, \ldots)$. We obtain a contradiction: The above set $S(T) = S$ is not a minimal dominant set for the uncertain problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ under consideration since $S(T)$ is not a minimal set with respect to inclusion (it is possible to remove any permutation $\pi_k \in S \setminus S'(T) \neq \emptyset$ from set $S(T) = S$, and the remaining set $S(T) \setminus \{\pi_k\}$ still satisfies Definition 1). Theorem 4 is proven. ■

The above proof of Theorem 4 implies the following claim.

Corollary 3. *Let $p_i^L < p_i^U, J_i \in \mathcal{J}$. For any permutation $\pi_k \in S$ there exists a vector $p \in T$ such that permutation π_k is the unique optimal permutation for the problem $1 \parallel \sum w_iC_i$ associated with the vector p of the job processing times if and only if inequality (15) holds.*

Proof. The proof of the *sufficiency* of condition (15) in Corollary 3 is entirely contained in that of Theorem 4. For a proof of the *necessity* of condition (15) in Corollary 3, we note that if for any permutation $\pi_k \in S$ there exists a vector $p \in T$ such that permutation π_k is the unique optimal permutation for the deterministic problem $1 \parallel \sum w_iC_i$ associated with the vector p of the job processing times, then (due to Definition 1) equality $S(T) = S$ holds for any minimal dominant set $S(T)$ constructed for the uncertain problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$. Hence, $|S(T)| = n!$ and the necessity of condition (15) in Corollary 3 follows from the necessity of condition (15) in Theorem 4. ■

It takes $O(n)$ time to check the condition (15) of Theorem 4 since $O(n)$ is the complexity of finding a maximum (minimum) in the set $\{\frac{w_i}{p_i^U} \mid J_i \in \mathcal{J}\}$ of real numbers (set $\{\frac{w_i}{p_i^L} \mid J_i \in \mathcal{J}\}$, respectively).

Table 1
Input data for Example 1.

i	p_i^L	p_i^U	w_i	$\frac{w_i}{p_i^L}$	$\frac{w_i}{p_i^U}$
1	5	6	30	6	5
2	4	8	8	2	1
3	1	3	18	18	6
4	3	4	12	4	3
5	4	5	20	5	4

Table 2
Data for Example 2.

i	p_i^L	p_i^U	w_i	$\frac{w_i}{p_i^L}$	$\frac{w_i}{p_i^U}$	$\frac{w_i}{p_i^L}+\frac{w_i}{p_i^U}$	$\frac{w_i}{p_i^L}\cdot\frac{w_i}{p_i^U}$
1	5	6	30	6	5	11	30
2	4	6	24	6	4	10	24
3	6	14	42	7	3	10	21
4	2	7	14	7	2	9	14
5	10	14	70	7	5	12	35

6. Illustrative examples

One can check the conditions of Theorems 2–4 in polynomial time. However, only Theorem 2 is of direct practical importance: If the condition (9) holds, then it is possible to construct an optimal permutation for the uncertain problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$, which remains optimal for any possible realization of the job processing times. Next, we demonstrate the applications of the theorems proven in Sections 4 and 5 by three illustrative examples.

6.1. Example 1

Let the input data for Example 1 of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ be given in columns 2–4 of Table 1. We test the condition (14) of Theorem 3 as follows:

$$\frac{w_3}{p_3^U}=6\ge 6=\frac{w_1}{p_1^L};\qquad \frac{w_1}{p_1^U}=5\ge 5=\frac{w_5}{p_5^L};\qquad \frac{w_5}{p_5^U}=4\ge 4=\frac{w_4}{p_4^L};\qquad \frac{w_4}{p_4^U}=3\ge 2=\frac{w_2}{p_2^L}.$$

Thus, for Example 1 of the uncertain problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$, the condition (14) holds. Hence due to Theorem 3, the minimal dominant set $S(T)$ for Example 1 is a singleton: $S(T)=\{\pi_k\}$, where $\pi_k=(J_3,J_1,J_5,J_4,J_2)$.

6.2. Example 2

We illustrate an application of Theorem 4 by Example 2 with the input data given in columns 2–4 of Table 2. We check the validity of condition (15):

$$\max\left\{\frac{w_i}{p_i^U}\mid J_i\in\mathcal{J}\right\}=5<6=\min\left\{\frac{w_i}{p_i^L}\mid J_i\in\mathcal{J}\right\}.$$

Therefore, for Example 2 of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ the condition (15) holds. Hence, according to Theorem 4, the minimal dominant set for this uncertain problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ has the maximal cardinality: $|S(T)|=5!=120$, i.e., a minimal dominant set $S(T)$ coincides with the set of all permutations of five jobs of set $\mathcal{J}$: $S(T)=S$.

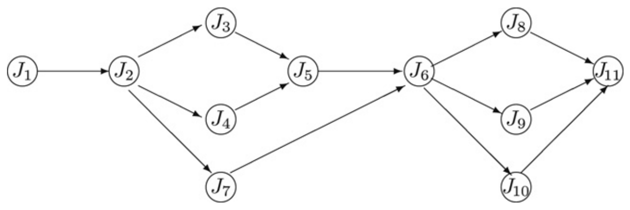
6.3. Example 3

We illustrate an application of Theorem 2 by Example 3. Let the input data for Example 3 be given in columns 2–4 of Table 3. For each pair of jobs $J_u\in\mathcal{J}$ and $J_v\in\mathcal{J}$ we check the validity of condition (9) of Theorem 2. The following relations hold:

$$\frac{w_1}{p_1^U}=6\ge 6=\frac{w_2}{p_2^L};\qquad \frac{w_2}{p_2^U}=5\ge 5=\frac{w_3}{p_3^L};\qquad \frac{w_2}{p_2^U}=5\ge 4=\frac{w_4}{p_4^L};\qquad \frac{w_3}{p_3^U}=2\ge 2=\frac{w_5}{p_5^L};$$
$$\frac{w_4}{p_4^U}=3\ge 2=\frac{w_5}{p_5^L};\qquad \frac{w_2}{p_2^U}=5\ge 5=\frac{w_7}{p_7^L};\qquad \frac{w_5}{p_5^U}=1\ge 1=\frac{w_6}{p_6^L};\qquad \frac{w_7}{p_7^U}=1.5\ge 1=\frac{w_6}{p_6^L};$$
$$\frac{w_6}{p_6^U}=0.5\ge 0.5=\frac{w_8}{p_8^L};\qquad \frac{w_6}{p_6^U}=0.5\ge 0.4=\frac{w_9}{p_9^L};\qquad \frac{w_6}{p_6^U}=0.5\ge 0.5=\frac{w_{10}}{p_{10}^L};$$
$$\frac{w_8}{p_8^U}=\frac{1}{3}\ge 0.3=\frac{w_{11}}{p_{11}^L};\qquad \frac{w_9}{p_9^U}=\frac{1}{3}\ge 0.3=\frac{w_{11}}{p_{11}^L};\qquad \frac{w_{10}}{p_{10}^U}=0.4\ge 0.3=\frac{w_{11}}{p_{11}^L}.$$

Table 3
Data for Example 3.

i	p_i^L	p_i^U	w_i	$\frac{w_i}{p_i^L}$	$\frac{w_i}{p_i^U}$	$\frac{w_i}{p_i^L}+\frac{w_i}{p_i^U}$	$\frac{w_i}{p_i^L}\cdot\frac{w_i}{p_i^U}$
1	1	3	18	18	6	24	84
2	5	6	30	6	5	11	30
3	4	10	20	5	2	7	10
4	3	4	12	4	3	7	12
5	4	8	8	2	1	3	2
6	10	20	10	1	0.5	1.5	0.5
7	3	10	15	5	1.5	6.5	7.5
8	10	15	5	0.5	$\frac{1}{3}$	$\frac{5}{6}$	$\frac{1}{6}$
9	5	6	2	0.4	$\frac{1}{3}$	$\frac{11}{15}$	$\frac{2}{15}$
10	20	26	10	0.5	$\frac{5}{13}$	0.5	$\frac{5}{26}$
11	10	20	3	0.3	0.15	0.45	$\frac{9}{200}$



$$\begin{tikzcd}
 & & J_3 \arrow[dr] & & & J_8 \arrow[dr] & \\
J_1 \arrow[r] & J_2 \arrow[ur] \arrow[dr] \arrow[ddr] & & J_5 \arrow[r] & J_6 \arrow[ur] \arrow[dr] \arrow[ddr] & & J_{11} \\
 & & J_4 \arrow[ur] & & & J_9 \arrow[ur] & \\
 & & J_7 \arrow[uurr] & & & J_{10} \arrow[uur] &
\end{tikzcd}$$

Fig. 1. Digraph $G^0 = (\mathcal{J}, \mathcal{A}^0)$ constructed for Example 3.

Thus, for Example 3 of the problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$, the condition (9) of Theorem 2 is satisfied for the following pairs of ordered jobs: (J_1,J_2), (J_2,J_3), (J_2,J_4), (J_3,J_5), (J_4,J_5), (J_2,J_7), (J_5,J_6), (J_7,J_6), (J_6,J_8), (J_6,J_9), (J_6,J_{10}), (J_8,J_{11}), (J_9,J_{11}), (J_{10},J_{11}). Hence due to Theorem 2 the following precedence–dominance relations hold: Job J_1 dominates job J_2; job J_2 dominates jobs J_3, J_4 and J_7; job J_3 dominates job J_5; job J_4 dominates job J_5; job J_5 dominates job J_6; job J_7 dominates job J_6; job J_6 dominates jobs J_8, J_9 and J_{10}; job J_8 dominates job J_{11}; job J_9 dominates job J_{11}; and job J_{10} dominates job J_{11}. No job from set $\{J_3,J_4,J_7\}$ dominates another one from this set. No job from set $\{J_5,J_7\}$ dominates another one from this set. No job from set $\{J_8,J_9,J_{10}\}$ dominates another one from this set. Thus, a minimal dominant set $S(T)$ for Example 3 is defined by the digraph $G^0 = (\mathcal{J}, \mathcal{A}^0)$ (Fig. 1) with the following set of non-transitive arcs: $\mathcal{A}^0 = \{(J_1,J_2), (J_2,J_3), (J_2,J_4), (J_2,J_7), (J_3,J_5), (J_4,J_5), (J_5,J_6), (J_7,J_6), (J_6,J_8), (J_6,J_9), (J_6,J_{10}), (J_8,J_{11}), (J_9,J_{11}), (J_{10},J_{11})\}$.

7. Robust scheduling based on the minimal dominant set

In this section, we present heuristic algorithms for solving problem $1|p_i^L \leq p_i \leq p_i^U|\sum w_iC_i$ based on Theorem 2 and on the optimality conditions derived for the two-job schedule minimizing the worst-case absolute regret (2) or relative regret (3). Similar conditions have been used in [8] for a branch-and-bound algorithm developed for the problem $1|p_i^L \leq p_i \leq p_i^U|\sum C_i$, where $w_i = 1$ for each job $J_i \in \mathcal{J}$.

If the condition (9) holds for jobs $J_u \in \mathcal{J}$ and $J_v \in \mathcal{J}$ or the symmetric condition $\frac{w_v}{p_v^U} \geq \frac{w_u}{p_u^L}$ holds, then the optimal ordering of jobs J_u and J_v may be obtained due to Theorem 2. Therefore, it is sufficient to consider only the case where both inequalities $\frac{w_u}{p_u^U} < \frac{w_v}{p_v^L}$ and $\frac{w_v}{p_v^U} < \frac{w_u}{p_u^L}$ hold. To order jobs J_u and J_v in the latter case we will use additional criterion of robust scheduling [8,9]. Permutation $\pi_k \in S$ is called *absolute* (*relative*) *robust*, if permutation π_k minimizes the worst-case absolute (relative) deviation from optimality, i.e., permutation π_k minimizes the value $Z(\pi_k)$ defined in (2) (value $Z'(\pi_k)$ defined in (3)) among permutations of set S.

It is easily verified that the worst-case for absolute or relative deviation is defined by processing times $p_u := p_u^U$ and $p_v := p_v^L$ for sequence (J_u,J_v), and by the processing times $p_u := p_u^L$ and $p_v := p_v^U$ for the opposite sequence (J_v,J_u). Thus, the worst-case absolute deviation from optimality for sequence (J_u,J_v) with respect to sequence (J_v,J_u) can be calculated as follows:

$$r((J_u,J_v),p_u^U,p_v^L) = \left(2\frac{w_u}{p_u^U} + \frac{w_v}{p_v^L}\right) - \left(2\frac{w_v}{p_v^L} + \frac{w_u}{p_u^U}\right) = \frac{w_u}{p_u^U} - \frac{w_v}{p_v^L}.$$

On the other hand, the worst-case absolute deviation from optimality for sequence (J_v,J_u) with respect to sequence (J_u,J_v) can be calculated as follows:

$$r((J_v,J_u),p_u^L,p_v^U) = \left(2\frac{w_v}{p_v^U} + \frac{w_u}{p_u^L}\right) - \left(2\frac{w_u}{p_u^L} + \frac{w_v}{p_v^U}\right) = \frac{w_v}{p_v^U} - \frac{w_u}{p_u^L}.$$

Therefore, sequence (J_u, J_v) is the absolute robust sequence with respect to sequence (J_u, J_v) if and only if inequality

$$\frac{w_u}{p_u^U} - \frac{w_v}{p_v^L} \le \frac{w_v}{p_v^U} - \frac{w_u}{p_u^L}$$

holds, or equivalently:

$$\frac{w_u}{p_u^L} + \frac{w_u}{p_u^U} \le \frac{w_v}{p_v^L} + \frac{w_v}{p_v^U}. \tag{17}$$

Similarly, the worst-case relative deviation from optimality for sequence (J_u, J_v) with respect to sequence (J_v, J_u) can be calculated as follows:

$$\frac{r((J_u, J_v), p_u^U, p_v^L)}{\gamma^*} = \frac{2\frac{w_u}{p_u^U} + \frac{w_v}{p_v^L}}{2\frac{w_v}{p_v^L} + \frac{w_u}{p_u^U}},$$

where γ^* denotes the minimal value of the objective function $\gamma = \sum_{i=1}^n w_i C_i$ for the best sequence of these two jobs, either (J_u, J_v) or (J_v, J_u), calculated for the actual job processing times. Inequalities $n \ge 2$, $w_i > 0$ and $p_i^L > 0$ for each job $J_i \in \mathcal{J}$ imply $\gamma^* > 0$. (We remind that the actual processing time of job $J_i \in \mathcal{J}$ becomes known only after completing job J_i.)

On the other hand, the worst-case relative deviation from optimality for sequence (J_v, J_u) with respect to sequence (J_u, J_v) can be calculated as follows:

$$\frac{r((J_v, J_u), p_u^L, p_v^U)}{\gamma^*} = \frac{2\frac{w_v}{p_v^U} + \frac{w_u}{p_u^L}}{2\frac{w_u}{p_u^L} + \frac{w_v}{p_v^U}}.$$

Therefore, sequence (J_u, J_v) is the relative robust sequence with respect to sequence (J_v, J_u) if and only if inequality

$$\frac{2\frac{w_u}{p_u^U} + \frac{w_v}{p_v^L}}{2\frac{w_v}{p_v^L} + \frac{w_u}{p_u^U}} \le \frac{2\frac{w_v}{p_v^U} + \frac{w_u}{p_u^L}}{2\frac{w_u}{p_u^L} + \frac{w_v}{p_v^U}},$$

holds or equivalently:

$$\frac{w_u}{p_u^L} \cdot \frac{w_u}{p_u^U} \le \frac{w_v}{p_v^L} \cdot \frac{w_v}{p_v^L}. \tag{18}$$

As a result, two heuristics for solving problem $1|p_i^L \le p_i \le p_i^U|\sum w_i C_i$ can be derived from the two-job optimality conditions represented by inequalities (17) and (18). The first heuristic called SUM computes sum

$$\frac{w_u}{p_u^L} + \frac{w_u}{p_u^U}$$

for each job $J_u \in \mathcal{J}$, and sorts set of jobs $\mathcal{J}$ in non-decreasing order of this sum.

The second heuristic called PROD computes product

$$\frac{w_u}{p_u^L} \cdot \frac{w_u}{p_u^U}$$

for each job $J_u \in \mathcal{J}$, and sorts set of jobs $\mathcal{J}$ in non-decreasing order of this product. Before presenting a formal algorithm for solving (exactly or heuristically) problem $1|p_i^L \le p_i \le p_i^U|\sum w_i C_i$, we demonstrate its procedure on Examples 1, 2, and 3.

7.1. Exact solution to Example 1

Testing the condition (9) of Theorem 2 for each pair of jobs from set $\mathcal{J}$ of Example 1 gives a completely ordered set of jobs: $J_3 \mapsto J_1 \mapsto J_5 \mapsto J_4 \mapsto J_2$.

Thus, permutation $\pi_k = (J_3, J_1, J_5, J_4, J_2) \in S$ provides an exact solution to Example 1 of the uncertain problem $1|p_i^L \le p_i \le p_i^U|\sum w_i C_i$. In other words, for any vector p of the possible processing times defined in columns 2–4 of Table 1, permutation π_k is optimal for the deterministic problem $1 \,\|\, \sum w_i C_i$ associated with the vector $p \in T$ of the job processing times.

7.2. Heuristic solutions to Example 2

Testing the condition (9) of Theorem 2 for each pair of jobs from set $\mathcal{J} = \{J_1, J_2, J_3, J_4, J_5\}$ in Example 2 gives the digraph $G = (\mathcal{J}, \mathcal{A}) = G^0$ with an empty set of arcs: $\mathcal{A} = \emptyset$. Thus, Example 2 is the most uncertain case of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_i C_i$ with five jobs and so one can only find a heuristic solution to Example 2.

Having applied the heuristic SUM for set $\mathcal{J}$, we obtain permutation $\pi_k = (J_4, J_2, J_3, J_1, J_5) \in S(T) = S$, which provides a heuristic solution to Example 2. Having applied the heuristic PROD for set $\mathcal{J}$, we obtain permutation $\pi_l = (J_4, J_3, J_2, J_1, J_5) \in S(T) = S$, which provides another heuristic solution to Example 2.

7.3. Heuristic solutions to Example 3

Testing the condition (9) of Theorem 2 for each pair of jobs from set $\mathcal{J} = \{J_1, J_2, J_3, J_4, J_5\}$ in Example 3 gives the digraph $G^0 = (\mathcal{J}, \mathcal{A}^0)$ representing a minimal dominant set $S(T)$ (see Fig. 1). Let P_i denote the list of predecessors of vertex J_i in the digraph G^0. We obtain $P_1 = \emptyset, P_2 = \{J_1\}, P_3 = \{J_1, J_2\}, P_4 = \{J_1, J_2\}, P_5 = \{J_1, J_2, J_3, J_4\}, P_6 = \{J_1, J_2, J_3, J_4, J_5, J_7\}, P_7 = \{J_1, J_2\}$, $P_8 = \{J_1, J_2, J_3, J_4, J_5, J_6, J_7\}, P_9 = \{J_1, J_2, J_3, J_4, J_5, J_6, J_7\}, P_{10} = \{J_1, J_2, J_3, J_4, J_5, J_6, J_7\}, P_{11} = \{J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8, J_9, J_{10}\}$. Since only job J_1 has an empty set P_1 of predecessors in the digraph G^0, job J_1 occupies the first position in the desired optimal permutation $\pi_k = (J_{k_1}, J_{k_2}, \dots, J_{k_{10}}) \in S(T)$. We fix the first position for job J_1 in permutation π_k as follows: $J_1 = J_{k_1}$, and delete job J_1 from all lists of predecessors. As a result, we obtain the following modified lists of predecessors: $P_i := P_i \setminus \{J_1\}$, $i \in \{2, 3, \dots, 11\}$.

Now, only job J_2 has an empty modified list of predecessors. Therefore, job J_2 occupies the second position in the desired optimal permutation $\pi_k \in S(T)$. We fix the second position for job J_2 in permutation π_k as follows: $J_2 = J_{k_2}$, and delete job J_2 from all the other lists of predecessors. As a result, we obtain the modified lists of predecessors: $P_i := P_i \setminus \{J_2\}$, $i \in \{3, 4, \dots, 11\}$.

Now, each of the three jobs J_3, J_4, and J_7 has an empty modified list of predecessors and we cannot find the optimal positions for jobs J_3, J_4, and J_7 in the desired optimal permutation $\pi_k \in T$. Let us order these three jobs using heuristic PROD. As a result, we obtain the sequence (J_7, J_3, J_4). (Note that the same sequence is obtained due to heuristic SUM.) Thus, job J_7 occupies the third position, job J_3 occupies the forth position, and job J_4 occupies the fifth position in permutation $\pi_k \in S(T)$. (Note that the above sequence (J_7, J_3, J_4) was not optimally chosen, and so permutation π_k may be not optimal for the actual processing times that are unknown before completing the corresponding jobs). We delete jobs J_3, J_4, and J_7 from all the other lists of predecessors and obtain the following modified lists of predecessors: $P_i := P_i \setminus \{J_3, J_4, J_7\}$, $i \in \{5, 6, 8, 9, 10, 11\}$.

Now, only job J_5 has an empty modified list of predecessors. Therefore, job J_5 occupies the sixth position in the desired optimal permutation $\pi_k \in S(T)$: $J_5 = J_{k_6}$. We delete job J_5 from all the other lists of predecessors, i.e., we obtain the following modified lists of predecessors: $P_i := P_i \setminus \{J_5\}$, $i \in \{6, 8, 9, 10, 11\}$.

Now, only job J_6 has an empty modified list of predecessors. Therefore, job J_6 occupies the seventh position in the desired optimal permutation $\pi_k \in S(T)$: $J_6 = J_{k_7}$. We delete job J_6 from all the other lists of predecessors, i.e., we obtain the following modified lists of predecessors: $P_i := P_i \setminus \{J_6\}$, $i \in \{8, 9, 10, 11\}$.

Now, each of three jobs J_8, J_9, and J_{10} has an empty modified list of predecessors and we cannot find the optimal positions of jobs J_8, J_9, and J_{10} in the desired optimal permutation $\pi_k \in T$. We can order these three jobs using heuristic PROD and obtain sequence (J_9, J_8, J_{10}). Thus, due to the heuristic PROD, job J_9 occupies the eighth position, job J_8 occupies the ninth position, and job J_{10} occupies the tenth position in the permutation $\pi_k \in S(T)$.

Another sequence (J_9, J_{10}, J_8) can be obtained using heuristic SUM. Due to the heuristic SUM job J_9 occupies the eighth position, job J_{10} occupies the ninth position, and job J_8 occupies the tenth position in permutation $\pi_k \in S(T)$. In both cases, we delete jobs J_8, J_9, and J_{10} from the list of predecessors of job J_{11} and obtain the following modified list of predecessors: $P_{11} := P_{11} \setminus \{J_8, J_9, J_{10}\} = \emptyset$. As a result, job J_{11} occupies the last position in the desired optimal permutation.

Thus, having applied Theorem 2 and the heuristic PROD for unordered subsets of jobs from set $\mathcal{J}$, we obtain permutation $\pi_k = (J_1, J_2, J_7, J_3, J_4, J_5, J_6, J_9, J_8, J_{10}, J_{11}) \in S(T)$, which provides a heuristic solution to Example 3. On the other hand, having applied Theorem 2 and the heuristic SUM for unordered subsets of jobs from set $\mathcal{J}$, we obtain permutation $\pi_l = (J_1, J_2, J_7, J_3, J_4, J_5, J_6, J_9, J_{10}, J_8, J_{11}) \in S(T)$, which provides another heuristic solution to Example 3. It should be reminded that before completing all jobs $\mathcal{J}$, one cannot calculate the exact values $\sum_{i=1}^{n} w_i C_i(\pi_k, p)$ and $\sum_{i=1}^{n} w_i C_i(\pi_l, p)$ of the objective function $\gamma = \sum_{i=1}^{n} w_i C_i$ since the vector $p \in T$ remains unknown: The actual vector $p^* \in T$ of job processing times will be known after completing the last job from set $\mathcal{J}$.

7.4. Algorithms for solving problem $1|p_i^L \le p_i \le p_i^U| \sum w_i C_i$

We will use the following notations in the formal description of the above algorithm used for solving (exactly or heuristically) Examples 1, 2, and 3.

Let π_k^m denote the subsequence of the first m jobs (with $1 \le m \le n$) in the desired permutation $\pi_k \in S(T)$: $\pi_k^1 = (J_{k_1})$, $\pi_k^2 = (J_{k_1}, J_{k_2}), \dots, \pi_k^n = \pi_k$.

π_k^0 denotes an empty subsequence of permutation π_k.

J denotes the subset of jobs, $J \subseteq \mathcal{J}$, still undetermined in the constructed subsequence π_k^m of the desired permutation π_k.

C denotes the subset of jobs of set J, $C \subseteq J$, without predecessors in the set J.

The following Algorithm $S(T)\&SUM$ is based on Theorem 2 and the heuristic SUM for the unordered subsets of vertices from set $\mathcal{J}$ in digraph G^0.

Algorithm $S(T)\&SUM$

```
Input:   Segments $[p_i^L, p_i^U]$ and weights $w_i$ for all jobs $J_i \in \mathcal{J}$.
Output:  Permutation $\pi_k \in S(T) \subseteq S$ providing a heuristic solution
         to the problem $1|p_i^L \le p_i \le p_i^U| \sum w_i C_i$.
  Step 1: Set $m := 0$ and $J := \mathcal{J}$;
```

```
        **FOR** $i = 1$ to $n$ **DO**
          set $P_i := \emptyset$;
        **END FOR**
step 2: **FOR** $u = 1$ to $n$ **DO**
          **FOR** $v = 1$ to $n$, $v \neq u$, **DO**
            test condition (9) for the pair of jobs $J_u \in \mathcal{J}$ and $J_v \in \mathcal{J}$;
            **IF** condition (9) holds **THEN** set $P_v := P_v \cup \{J_u\}$;
          **END FOR**
        **END FOR**
step 3: Set $C := \emptyset$;
        **FOR** $J_i \in J$ **DO**
          **IF** $P_i = \emptyset$ **THEN** set $C := C \cup \{J_i\}$;
        **END FOR**
step 4: **IF** $|C| = 1$ **THEN** set $\pi_k^{m+1} := (J_{k_1}, J_{k_2}, \dots, J_{k_m}, J_r)$,
          where $\pi_k^m = (J_{k_1}, J_{k_2}, \dots, J_{k_m})$ and $C = \{J_r\}$;
          **FOR** $J_i \in J$ **DO**
            set $P_i := P_i \setminus \{J_r\}, J := J \setminus \{J_r\}$;
          **END FOR**
          set $m := m + 1$ **GOTO** step 3;
        **ELSE**
step 5: **FOR** $J_i \in C$ **DO**
          compute $h(J_i) = \frac{w_i}{p_i^L} + \frac{w_i}{p_i^U}$;
        **END FOR**
        create sequence $(J_{i_1}, J_{i_2}, \dots, J_{i_{|C|}})$ by sorting all jobs in set $C$
        in non-decreasing order of the values $h(J_{i_t}),\ t \in \{1, 2, \dots, |C|\}$.
step 6: Set $\pi_k^{m+|C|} := (J_{k_1}, J_{k_2}, \dots, J_{k_m}, J_{i_1}, J_{i_2}, \dots, J_{i_{|C|}})$,
        where $\pi_k^m = (J_{k_1}, J_{k_2}, \dots, J_{k_m})$;
        **IF** $m + |C| = n$ **THEN STOP**
        **ELSE**
        **FOR** $J_i \in J$ **DO**
          set $P_i := P_i \setminus C, J := J \setminus C$;
        **END FOR**
        set $m := m + |C|$ **GOTO** step 3.
```

To obtain Algorithm $S(T)\&PROD$ based on Theorem 2 and the heuristic PROD, it is sufficient to substitute equality

$$h(J_i) = \frac{w_i}{p_i^L} + \frac{w_i}{p_i^U} \quad \text{by equality } h(J_i) = \frac{w_i}{p_i^L} \cdot \frac{w_i}{p_i^U}$$

at step 5 of Algorithm $S(T)\&SUM$. If either the condition (17) or condition (18) turns out to be an equality, then job J_u precedes job J_v in permutation $\pi_k \in S(T)$ while the other condition (18) (condition (17), respectively) holds as strict inequality. If both conditions (17) and (18) turn out to be equalities and $u < v$, then job J_u precedes job J_v in permutation $\pi_k \in S(T)$ constructed by heuristic SUM (by heuristic PROD).

Both Algorithm $S(T)\&SUM$ and Algorithm $S(T)\&PROD$ take $O(n^2)$ time. This complexity is defined by step 2. In the general case of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ Algorithm $S(T)\&SUM$ (Algorithm $S(T)\&SUM$, respectively) generates a heuristic solution for absolute (relative) robust scheduling. However, there is a special case of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ in which each of these algorithms provides an exact solution for robust scheduling.

Let the set of jobs $\mathcal{J}$ in the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ be decomposed into $r \le n$ subsets $\mathcal{J} = \mathcal{J}^1 \cup \mathcal{J}^2 \cup \dots \cup \mathcal{J}^r$ such that the following conditions hold:

(i) $|\mathcal{J}^k| \le 2$ for each $k \in \{1, 2, \dots, r\}$;
(ii) precedence–dominance relation $J_u \mapsto J_v$ holds for each pair of jobs $J_u \in \mathcal{J}^k$ and $J_v \in \mathcal{J}^s$ with $k < s$.

Proposition 1. *If for the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ both conditions* (i) *and* (ii) *hold, then Algorithm $S(T)\&SUM$ (Algorithm $S(T)\&PROD$, respectively) generates an absolute (relative) robust permutation for this problem.*

Proof. Due to conditions (i) and (ii), at each iteration of Algorithm $S(T)\&SUM$ (Algorithm $S(T)\&PROD$) the cardinality of set C is not greater than 2. If $|C| = 1$, then the position of job $J_r \in J \in \mathcal{J}$ in the constructed permutation $\pi_k \in S(T)$ is chosen in an optimal way due to Theorem 2. If $|C| = 2$, then sequence (J_{i_1}, J_{i_2}) of jobs $J_{i_1} \in C$ and $J_{i_2} \in C$ is the absolute (relative) robust sequence with respect to the opposite sequence (J_{i_2}, J_{i_1}) due to the sufficient condition (17) (the sufficient condition (18), respectively). Thus, each local decision for selecting a job from the set $\mathcal{J}$ to be processed next is realized in an optimal

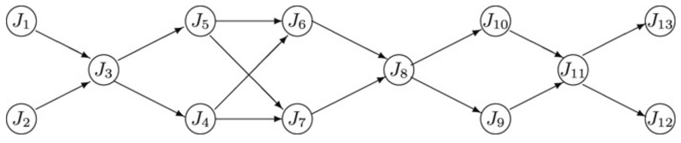


Fig. 2. Digraph $G^0 = (\mathcal{J}, \mathcal{A}^0)$ constructed for Example 4.

way (in the sense of robustness). Therefore, the final permutation obtained by Algorithm $S(T)\&SUM$ (Algorithm $S(T)\&SUM$) is an absolute (relative) robust permutation for the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ due to the additivity of the objective function $\gamma = \sum_{i=1}^{n} w_iC_i$. ■

It is easy to convince that for Example 4 of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ with the digraph $G^0 = (\mathcal{J}, \mathcal{A}^0)$ represented in Fig. 2 both conditions (i) and (ii) of Proposition 1 hold. Therefore, Algorithm $S(T)\&SUM$ (Algorithm $S(T)\&SUM$, respectively) generates an absolute (relative) robust permutation for Example 4. Note that the level of the uncertainty (with respect to Definition 1) of Example 4 may be evaluated as follows: $|S(T)| = 2^5 = 32$.

8. Computational results

Via the testing of randomly generated instances of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ we answer (by experiments on PC) the question of how many pairs of jobs from the set $\mathcal{J}$ satisfies the condition (9) of Theorem 2. We estimate how many randomly generated instances satisfies condition (14) of Theorem 3, condition (15) of Theorem 4, and both conditions (i) and (ii) of Proposition 1. We estimate also how large the relative error Δ (in percentage) of the value $\gamma_{p^*}^k$ of the objective function $\gamma = \sum_{i=1}^{n} w_iC_i$ is obtained for the schedule π_k constructed due to Algorithm $S(T)\&SUM$ (due to Algorithm $S(T)\&PROD$) with respect to the actually optimal objective value $\gamma_{p^*}^t$ calculated for the actual processing times $p^* = (p_1^*, p_2^*, \dots, p_n^*) \in T$:

$$\Delta = \frac{\gamma_{p^*}^k - \gamma_{p^*}^t}{\gamma_{p^*}^t} \cdot 100\%. \tag{19}$$

The actual processing times $p_i^*, J_i \in \mathcal{J}$, are assumed to be unknown before constructing a schedule π_k by Algorithm $S(T)\&SUM$ (by Algorithm $S(T)\&PROD$). In fact, an actual processing time p_i^* was uniformly distributed in the range $[p_i^L, p_i^U]$. And the actually optimal permutation $\pi_t \in S$ defining the optimal objective value $\gamma_{p^*}^t$ for the deterministic problem $1 \,\|\, \sum w_iC_i$ associated with the actual job processing times $p_i^*, J_i \in \mathcal{J}$, was calculated using the optimality condition (4) of Theorem 1.

For the computational experiments, we used a Pentium IV with 2.80 GHz processor and 248 MB main memory. Generator from [24] has been used for (pseudo) random generating instances of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$. Algorithm $S(T)\&SUM$ and Algorithm $S(T)\&PROD$ have been coded in C++.

Note that if Algorithm $S(T)\&SUM$ (Algorithm $S(T)\&PROD$) detects a precedence relation $J_u \mapsto J_v$, i.e., inclusion $(J_u, J_v) \in \mathcal{A}$ holds (provided that $u < v$), then the algorithm does not test the validity of the opposite relation: $J_v \mapsto J_u$, i.e., $(J_v, J_u) \notin \mathcal{A}$. Otherwise, if Algorithm $S(T)\&SUM$ (Algorithm $S(T)\&PROD$) detects that the precedence relation $J_u \mapsto J_v$ does not hold (and $u < v$), then the algorithm tests the validity of the opposite relation: $J_v \mapsto J_u$. In this case, if relation $J_v \mapsto J_u$ holds, then $(J_v, J_u) \in \mathcal{A}$ and $(J_u, J_v) \notin \mathcal{A}$. Otherwise, $(J_v, J_u) \notin \mathcal{A}$ and $(J_u, J_v) \notin \mathcal{A}$. Consequently, if equality

$$|\mathcal{A}| = \frac{n(n-1)}{2} \tag{20}$$

holds, then Algorithm $S(T)\&SUM$ (Algorithm $S(T)\&PROD$) constructs a permutation $\pi_k \in S(T)$ that is optimal for any vector $p \in T$ of job processing times without fail, i.e., equality $\gamma_{p^*}^k = \gamma_{p^*}^t$ must hold. Indeed, in such a case if equality (20) holds, then digraph G turns out to be a tournament (a complete circuit-free digraph), and so the condition (15) of Theorem 4 necessarily holds.

Tables 4 and 5 represent the computational results of testing randomly generated instances of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ with $n \in \{5, 10, 25, 50, 75, 100, 200, 400, 700, 1000\}$. In fact, we tested series of instances for integer n from the range $[5, 100]$ with step 5 and for integer n from the range $[100, 1000]$ with step 50 (among others). However for brevity, we did not present the intermediate results in Tables 4 and 5 since they do not change the general sense of the results of the experiments. An integer lower bound p_i^L and integer upper bound p_i^U of the possible real values $p_i \in R_+$ of the job processing times, $p_i \in [p_i^L, p_i^U]$, have been generated as follows. First, an integer center C of the closed interval $[p_i^L, p_i^U]$ was generated using a uniform distribution in the given range $[L, U]$: $L \le C \le U$. Then the lower bound p_i^L of a possible processing time was defined using equality $p_i^L = C \cdot (1 - \frac{\delta}{100})$. And an upper bound p_i^U was defined using equality $p_i^U = C \cdot (1 + \frac{\delta}{100})$. As a result, a maximal possible relative error of the uncertain processing time was equal to $2\delta\%$.

Table 4
Randomly generated instances with $[L, U] = [1, 200]$, $w_i \in [1, 50]$ and $\delta \in \{0.1, 0.5, 1, 5, 10\}$.

	n	δ (%)	$\vert\mathcal{A}\vert$ (%)	Error Δ for $S(T)$&*SUM*	Error Δ for $S(T)$&*PRO*	$\vert S(T)\vert = 1$	$\vert S(T)\vert = n!$	$Z(\pi_k) = 0$	CPU-time (s)
1	2	3	4	5	6	7	8	9	10
1	5	0.1	100	0.000000	0.000000	10	0	0	0
2	10	0.1	100	0.000000	0.000000	10	0	0	0
3	25	0.1	99.800000	0.000346	0.000346	7	0	0	0
4	50	0.1	99.910204	0.000054	0.000054	3	0	0	0
5	75	0.1	99.909910	0.000110	0.000110	1	0	0	0
6	100	0.1	99.890909	0.000027	0.000027	0	0	0	0.1
7	200	0.1	99.909045	0.000072	0.000072	0	0	0	0.3
8	400	0.1	99.890100	0.000172	0.000172	0	0	0	3.3
9	700	0.1	99.902923	0.000149	0.000150	0	0	0	37.5
10	1000	0.1	99.895776	0.000156	0.000156	0	0	0	226.4
11	5	0.5	99.000000	0.000000	0.000000	9	0	0	0
12	10	0.5	99.777778	0.000000	0.000000	9	0	0	0
13	25	0.5	99.533333	0.000731	0.000731	2	0	0	0
14	50	0.5	99.420408	0.001109	0.001109	0	0	0	0
15	75	0.5	99.473874	0.000563	0.000563	0	0	0	0
16	100	0.5	99.466667	0.000679	0.000679	0	0	0	0
17	200	0.5	99.481407	0.000604	0.000604	0	0	0	0.2
18	400	0.5	99.500125	0.000619	0.000622	0	0	0	3.3
19	700	0.5	99.479542	0.000667	0.000666	0	0	0	44.2
20	1000	0.5	99.479600	0.000659	0.000658	0	0	0	209.6
21	5	1	98.000000	0.000000	0.000000	8	0	0	0
22	10	1	99.333333	0.000398	0.000398	7	0	0	0
23	25	1	98.700000	0.000616	0.000616	1	0	0	0
24	50	1	99.085714	0.001731	0.001731	0	0	0	0
25	75	1	99.012613	0.001632	0.001632	0	0	0	0
26	100	1	99.010101	0.002142	0.002142	0	0	0	0
27	200	1	98.923618	0.001992	0.001992	0	0	0	0.2
28	400	1	98.937719	0.002084	0.002073	0	0	0	3.2
29	700	1	98.967668	0.002089	0.002086	0	0	0	38.6
30	1000	1	98.953554	0.002085	0.002081	0	0	0	220.8
31	5	5	91.000000	0.129772	0.129772	4	0	2	0
32	10	5	95.111111	0.084177	0.084177	3	0	0	0
33	25	5	95.033333	0.041485	0.041485	0	0	0	0
34	50	5	94.546939	0.054941	0.054941	0	0	0	0
35	75	5	94.781982	0.044395	0.044254	0	0	0	0
36	100	5	94.632323	0.044177	0.044327	0	0	0	0
37	200	5	94.765829	0.048270	0.048372	0	0	0	0.2
38	400	5	94.854762	0.048875	0.048896	0	0	0	2.9
39	700	5	94.790170	0.046831	0.046897	0	0	0	38.9
40	1000	5	94.790170	0.047052	0.047036	0	0	0	199.7
41	5	10	90.000000	0.114561	0.114561	2	0	1	0
42	10	10	90.444444	0.154431	0.154431	0	0	0	0
43	25	10	89.533333	0.199577	0.199577	0	0	0	0
44	50	10	88.873469	0.153312	0.153312	0	0	0	0
45	75	10	90.061261	0.180604	0.181736	0	0	0	0
46	100	10	89.664646	0.204054	0.203940	0	0	0	0
47	200	10	89.246231	0.193124	0.192861	0	0	0	0.1
48	400	10	89.685088	0.186289	0.186374	0	0	0	2.5
49	700	10	89.483630	0.190972	0.191143	0	0	0	24.5
50	1000	10	89.559079	0.190070	0.190204	0	0	0	168.8

In the experiments, we tested instances of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ with the relative errors $2\delta\%$ of the random processing times defined by the following values of $\delta \in \{0.1, 0.5, 1.0, 5.0, 10.0, 15.0, 25.0, 50.0, 75.0, 100.0\}$. Table 4 represents the computational results for small relative errors of the job processing times: $\delta \in \{0.1, 0.5, 1.0, 5.0, 10.0\}$, while Table 5 represents the computational results for large relative errors of the job processing times: $\delta \in \{15.0, 25.0, 50.0, 75.0, 100.0\}$. In both Tables 4 and 5, the same range $[L, U]$ for varying center C of the closed interval $[p_i^L, p_i^U]$ was used, namely: $L = 1$ and $U = 200$. For each job $J_i \in \mathcal{J}$, the real weight $w_i \in R_+$ was uniformly distributed in the same range $[1, 50]$. Of course, the weight w_i is assumed to be known exactly before scheduling (in contrast to job processing time p_i which is assumed to be unknown before completion time C_i).

Tables 4 and 5 represent the computational results for 100 series of the randomly generated instances of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$. Each series includes 10 instances with the same combination of n and δ. The series number is given in column 1. The number n of jobs in an instance is given in column 2. The half of the maximal possible error δ of the random processing times (in percentage) is given in column 3. Column 4 represents the average relative number $|\mathcal{A}|$ of arcs

Table 5
Randomly generated instances with $[L, U] = [1, 200]$, $w_i \in [1, 50]$ and $\delta \in \{15, 25, 50, 75, 100\}$.

	n	δ (%)	$\lvert\mathcal{A}\rvert$ (%)	Error Δ for $S(T)\&SUM$	Error Δ for $S(T)\&PRO$	$\lvert S(T)\rvert = 1$	$\lvert S(T)\rvert = n!$	$Z(\pi_k) = 0$	CPU-time (s)
1	2	3	4	5	6	7	8	9	10
51	5	15	90.000000	0.464456	0.464456	4	0	2	0
52	10	15	85.555556	0.259672	0.259672	0	0	0	0
53	25	15	85.466667	0.358869	0.361898	0	0	0	0
54	50	15	84.269388	0.385431	0.385431	0	0	0	0
55	75	15	84.079279	0.364014	0.363045	0	0	0	0
56	100	15	84.947475	0.399651	0.399766	0	0	0	0.1
57	200	15	84.289447	0.398092	0.398966	0	0	0	0.2
58	400	15	84.044236	0.421430	0.422301	0	0	0	2.1
59	700	15	84.508441	0.415488	0.415682	0	0	0	26.3
60	1000	15	84.467067	0.417893	0.417752	0	0	0	141.7
61	5	25	78.000000	0.923429	0.923429	0	0	0	0
62	10	25	79.555556	0.223605	0.223605	0	0	0	0
63	25	25	74.866667	0.911780	0.911780	0	0	0	0
64	50	25	73.991837	1.037238	1.038106	0	0	0	0
65	75	25	73.545946	1.092398	1.095871	0	0	0	0
66	100	25	74.135354	1.165435	1.164930	0	0	0	0
67	200	25	74.912060	1.188814	1.188926	0	0	0	0.1
68	400	25	74.483960	1.162628	1.162578	0	0	0	1.6
69	700	25	74.243205	1.200619	1.200902	0	0	0	15.1
70	1000	25	73.998458	1.207374	1.206891	0	0	0	94.9
71	5	50	51.000000	0.737772	0.737772	0	0	0	0
72	10	50	44.222222	3.906967	3.934715	0	0	0	0
73	25	50	47.566667	4.718102	4.658379	0	0	0	0
74	50	50	51.134694	3.829368	3.843392	0	0	0	0
75	75	50	45.924324	5.013883	5.017818	0	0	0	0
76	100	50	47.822222	4.806553	4.792812	0	0	0	0
77	200	50	49.427638	4.740232	4.745235	0	0	0	0.1
78	400	50	49.316917	4.929315	4.926666	0	0	0	0.6
79	700	50	49.085469	5.149548	5.151449	0	0	0	5.3
80	1000	50	49.439179	5.008964	5.009273	0	0	0	27.8
81	5	75	31.000000	4.354902	4.354902	0	2	0	0
82	10	75	18.666667	5.992706	5.992706	0	1	0	0
83	25	75	25.100000	11.896708	11.914977	0	0	0	0
84	50	75	25.183673	11.070877	11.024420	0	0	0	0
85	75	75	28.580180	11.075583	11.078941	0	0	0	0
86	100	75	23.569697	10.408500	10.411961	0	0	0	0
87	200	75	25.677889	11.435902	11.437732	0	0	0	0
88	400	75	25.797870	11.308924	11.318648	0	0	0	0.1
89	700	75	25.222481	12.108815	12.113248	0	0	0	1.1
90	1000	75	25.445586	11.837250	11.845775	0	0	0	5.1
91	5	100	1.000000	13.576332	23.720536	0	9	0	0
92	10	100	0.444444	19.100562	27.841165	0	9	0	0
93	25	100	0.600000	21.785152	49.358878	0	5	0	0
94	50	100	0.848980	26.782155	60.921893	0	2	0	0
95	75	100	0.893694	29.136338	56.236835	0	0	0	0
96	100	100	0.894949	27.903204	53.457656	0	0	0	0
97	200	100	0.356281	29.813504	57.447367	0	0	0	0
98	400	100	0.562155	27.955208	55.066314	0	0	0	0
99	700	100	0.468056	28.982720	52.711515	0	0	0	0
100	1000	100	0.480160	28.709586	53.917875	0	0	0	0.1

in the digraph $G = (\mathcal{J}, \mathcal{A})$ constructed using the condition (9) of Theorem 2 (in percentage of the arc number in a complete circuit-free digraph of order n):

$$|\mathcal{A}| : \left(\frac{n(n-1)}{2}\right) \cdot 100\%.$$

The average relative error Δ of the objective function value $\gamma^k_{p^*}$ calculated for permutation π_k constructed by Algorithm $S(T)\&SUM$ (by Algorithm $S(T)\&PROD$) with respect to the optimal objective function value $\gamma^t_{p^*}$ defined for actual job processing times is given in columns 5 (in column 6, respectively). Definition of Δ is given by equality (19). Column 7 represents the number of instances (from 10 ones in a series) for which the condition (14) of Theorem 3 holds, i.e., in such an instance there exists a single dominant permutation $\pi_k \in S$ that is optimal for any possible vector $p \in T$ of job processing times: $|S(T)| = 1$. Column 8 represents the number of instances (from 10 ones in a series) for which the condition (15) of Theorem 4 holds, i.e., $|S(T)| = n!$. Column 9 represents the number of instances (from 10 ones in a series) for which both conditions (i) and (ii) of Proposition 1 hold, i.e., permutation π_k constructed by Algorithm $S(T)\&SUM$ (by Algorithm $S(T)\&PROD$) is an absolute (relative) robust permutation for the considered instance of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$. The average CPU-time (in seconds) used by the 2.80 GHz processor for solving one instance (exactly of heuristically) and for conducting the whole of the above analysis is given in columns 10.

From the experiments, it follows that the condition (14) of Theorem 3 holds only for some instances with small number of jobs ($5 \le n \le 75$) and rather small relative error $2\delta\%$ of job processing times (see column 7 for series with numbers 1–5, 11–13, 21–23, 31, 32, 41, 51). Moreover, only two series of instances (with numbers 1 and 2) were completely solved (i.e., $\gamma_{p^*}^k = \gamma_{p^*}^t$) due to the validity of condition (14). Both conditions (i) and (ii) of Proposition 1 hold only for two instances from the series with number 51 (see column 8). The condition (15) of Theorem 4 holds only for a few instances with a small number of jobs ($5 \le n \le 50$) and large relative error ($\delta \in \{75, 100\}$) of job processing times (see column 8 for series with numbers 81, 82, 91–94). Note that a lot of pairs of jobs from set $\mathcal{J}$ satisfy the condition (9) of Theorem 2. Due to the large number of arcs $\mathcal{A}$ (see column 4) each of the both Algorithms $S(T)\&SUM$ and $S(T)\&PROD$ generate a permutation π_k with very low average relative error Δ of the actually optimal value $\gamma_{p^*}^t$ of the objective function (see columns 5 and 6 for series with numbers 1–82). In particular, for instance series 1–63, the average values Δ are less than 1%. For instance series 64–78, average values Δ are less than 5%. The average quality of the schedules obtained depends of the error $2\delta\%$ of the job processing times and remains rather close for the instances with different number of jobs provided that they have the same δ of the uncertain processing times. For $\delta \in \{0.1, 0.5, 1.0, 5.0, 10.0, 15.0, 25.0, 50.0, 75.0\}$, the quality of solutions obtained by Algorithm $S(T)\&SUM$ is closed to that obtained by Algorithm $S(T)\&PROD$ while for $\delta = 100$ when only a few arcs are in set $\mathcal{A}$ (see column 4), the quality of solutions obtained by Algorithm $S(T)\&SUM$ is essentially better than that obtained by Algorithm $S(T)\&PROD$.

The lowest average relative number of arcs $\mathcal{A}$, i.e., 0.444444%, was constructed for the instance series with number 92. The lowest average solution quality, i.e., $\Delta = 60.921893\%$, was obtained by Algorithm $S(T)\&PROD$ for the series with number 94.

The CPU-time depends both on the number of jobs n and on the number of arcs $\mathcal{A}$. The latter dependence is provided by testing the conditions (i) and (ii) of Proposition 1. For example, the largest average CPU-time, 226.4 s, is obtained for the series with number 10 since the number of jobs is the largest, $n = 1000$, and the number of arcs is large, 99.895776, as well.

9. Concluding remarks

For the uncertain problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$, the necessary and sufficient condition is proven for the existence of a single permutation of the n jobs which remains optimal for all possible processing times (the simplest case of an uncertain problem). The necessary and sufficient condition is also proven for the case that for any permutation of the n jobs there exists such a vector of possible processing times that this permutation is uniquely optimal (the hardest case of an uncertain problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$). Both the necessary and sufficient conditions proven may be tested in polynomial time of the number n of jobs. For the general case of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$, the precedence–dominance relations are developed. If there is no precedence–dominance relation within some jobs, the worst-case regret criterion was used for sequencing such a subset of jobs. A special case of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ is identified when a permutation minimizing the absolute (relative) worst-case regret may be calculated in polynomial time.

The main issue of this paper is to show how to use a minimal dominant set of permutations $S(T)$, $S(T) \subseteq S$, for an optimal realization of the process in a special case of the problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ and for minimizing the worst-case regret. We estimate the strength of using a minimal dominant set $S(T)$ by extensive computational experiments for randomly generated problems $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ with the number n of jobs from the range [5, 1000]. In particular, it was shown by experiments on PC that by using a minimal dominant set one can find a permutation $\pi_k \in S(T)$ with a rather small relative error Δ of the actually optimal value $\gamma_{p^*}^k$ of the objective function $\gamma = \sum_{i=1}^n w_iC_i$.

The obtained results may be used in the hierarchical method to scheduling adopted over the last decade and corresponds to industrial practices [2]. In the *static phase* (off-line scheduling), when the level of uncertainty of the input data is high, a scheduler can find a minimal dominant set $S(T)$ to the uncertain problem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$. In order to find an optimal schedule, the subset of schedules from the set $S(T)$ to the uncertain subproblem $1|p_i^L \le p_i \le p_i^U|\sum w_iC_i$ obtained after each decision being made in the *dynamic phase* (on-line scheduling) has to remain a minimal dominant one for the remaining subset of jobs. In [25], it was shown by experiments that such an hierarchical approach based on a minimal dominant set $S(T)$ is very efficient for the uncertain two-machine flow-shop problem $F2|p_i^L \le p_i \le p_i^U|C_{\max}$ with $C_{\max}$ criterion.

Acknowledgements

The authors would like to thank to Professor N. Bellomo and an anonymous referee for their comments on the earlier draft of the paper and suggestions. The research of the first author was supported by Belarusian Republican Foundation for Fundamental Research (grant M08MC-027) and by National Science Council of Taiwan (grant 98-2811-H-002-005). The research of the third author was supported by National Science Council of Taiwan (grant 97-2410-H-002-107).

References

[1] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Prentice-Hall, USA, Englewood Cliffs, 1995.
[2] H. Aytug, M.A. Lawley, K. McKay, S. Mohan, R. Uzsoy, Executing production schedules in the face of uncertainties: A review and some future directions, European Journal of Operational Research 161 (2005) 86–110.
[3] A. Barbagallo, Regularity results for time-dependent variational and quasi-variational inequalities and application to the calculation of dynamic traffic network, Mathematical Models and Methods in Applied Sciences 17 (2) (2007) 277–304.
[4] T.-C. Lai, Yu.N. Sotskov, N. Sotskova, F. Werner, Optimal makespan scheduling with given bounds of processing times, Mathematical and Computer Modelling 26 (1997) 67–86.
[5] T.-C. Lai, Yu.N. Sotskov, Sequencing with uncertain numerical data for makespan minimization, Journal of the Operational Research Society 50 (1999) 230–243.
[6] M. Sevaux, K. Sorensen, A genetic algorithm for robust schedules in a one-machine environment with ready times and due dates, 4OR A Quarterly Journal of Operations Research 2 (2) (2004) 129–147.
[7] I. Averbakh, Minmax regret solutions for minmax optimization problems with uncertainty, Operations Research Letters 27 (2000) 57–65.
[8] R.L. Daniels, P. Kouvelis, Robust scheduling to hedge against processing time uncertainty in single-stage production, Management Science 41 (2) (1995) 363–376.
[9] P. Kouvelis, G. Yu, Robust Discrete Optimization and Its Applications, Kluwer Academic Publishers, Boston, The USA, 1997.
[10] V. Lebedev, I. Averbakh, Complexity of minimizing the total flow time with interval data and minmax regret criterion, Discrete Applied Mathematics 154 (2006) 2167–2177.
[11] J. Yang, G. Yu, On the robust single machine scheduling problem, Journal of Combinatorial Optimization 6 (2002) 17–33.
[12] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling. A survey, Annals of Discrete Mathematics 5 (1976) 287–326.
[13] W.E. Smith, Various optimizers for single-stage production, Naval Research and Logistics Quarterly 3 (1) (1956) 59–66.
[14] I. Averbakh, On the complexity of a class of combinatorial optimization problems with uncertainty, Mathematical Programming, Series A 90 (2001) 263–272.
[15] R. Leus, W. Herroelen, The complexity of machine scheduling for stability with a single disrupted job, Operations Research Letters 33 (2005) 151–156.
[16] R. Leus, W. Herroelen, Scheduling for stability in single-machine production systems, Journal of Scheduling 10 (2007) 223–235.
[17] P. Kouvelis, R.L. Daniels, G. Vairaktarakis, Robust scheduling of a two-machine flow shop with uncertain processing times, IIE Transactions 32 (2000) 421–432.
[18] A. Kasperski, Minimizing maximal regret in the single machine sequencing problem with maximum lateness criterion, Operations Research Letters 33 (2005) 431–436.
[19] A. Allahverdi, Yu.N. Sotskov, Two-machine flowshop minimum-length scheduling problem with random and bounded processing times, International Transactions in Operational Research 10 (2003) 65–76.
[20] A. Allahverdi, T. Aldowaisan, Yu.N. Sotskov, Two-machine flowshop scheduling problem to minimize makespan or total completion time with random and bounded setup times, International Journal of Mathematical Sciences 39 (2003) 2475–2486.
[21] N.M. Leshchenko, Yu.N. Sotskov, Realization of an optimal schedule for the two-machine flow-shop with interval processing times, International Journal Information Theories & Applications 14 (2007) 182–189.
[22] Yu.N. Sotskov, A. Allahverdi, T.-C. Lai, Flowshop scheduling problem to minimize total completion time with random and bounded processing times, Journal of the Operational Research Society 55 (2004) 277–286.
[23] E.G. Koffman (Ed.), Computer and Job-Shop Scheduling Theory, John Wiley & Sons, 1976.
[24] E. Taillard, Benchmarks for basic scheduling problems, European Journal of Operational Research 64 (1993) 278–285.
[25] N.M. Matsveichuk, Yu.N. Sotskov, N.G. Egorova, T.-C. Lai, Schedule execution for two-machine flow-shop with interval processing times, Mathematical and Computer Modelling 49 (5–6) (2009) 991–1011.