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

Article *in* Mathematical and Computer Modelling · August 2009

DOI: 10.1016/j.mcm.2009.03.006 · Source: DBLP

3 authors, including:

Yuri N. Sotskov
United Institute Of Informatics Problems

Tsung-Chyan Lai
Harbin Engineering University

Mathematical and Computer Modelling 49 (2009) 991–1011

Contents lists available at ScienceDirect

Mathematical and Computer Modelling

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

Schedule execution for two-machine flow-shop with interval processing times

N.M. Matsveichuk [a], 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 15 January 2008
Accepted 26 February 2008

Keywords:
Scheduling
Flow-shop
Makespan
Uncertainty
Dominant schedule

ABSTRACT

This paper addresses the issue of how to best execute the schedule in a two-phase scheduling decision framework by considering a two-machine flow-shop scheduling problem in which each uncertain processing time of a job on a machine may take any value between a lower and upper bound. The scheduling objective is to minimize the makespan. There are two phases in the scheduling process: the off-line phase (the schedule planning phase) and the on-line phase (the schedule execution phase). The information of the lower and upper bound for each uncertain processing time is available at the beginning of the off-line phase while the local information on the realization (the actual value) of each uncertain processing time is available once the corresponding operation (of a job on a machine) is completed. In the off-line phase, a scheduler prepares a minimal set of dominant schedules, which is derived based on a set of sufficient conditions for schedule domination that we develop in this paper. This set of dominant schedules enables a scheduler to quickly make an on-line scheduling decision whenever additional local information on realization of an uncertain processing time is available. This set of dominant schedules can also optimally cover all feasible realizations of the uncertain processing times in the sense that for any feasible realizations of the uncertain processing times there exists at least one schedule in this dominant set which is optimal. Our approach enables a scheduler to best execute a schedule and may end up with executing the schedule optimally in many instances according to our extensive computational experiments which are based on randomly generated data up to 1000 jobs. The algorithm for testing the set of sufficient conditions of schedule domination is not only theoretically appealing (i.e., polynomial in the number of jobs) but also empirically fast, as our extensive computational experiments indicate.

1. Introduction

There are two types of stochastic flow-shop scheduling problems traditionally addressed in the OR literature [1], where one is on stochastic job and the other is on stochastic machine. In a stochastic *job* problem, each job processing time is assumed to be a random variable following a certain probability distribution. With an objective of stochastically minimizing the makespan (i.e. minimizing the expected schedule length), the flow-shop problem was considered in articles [2–4], among others. In a stochastic *machine* problem, each job processing time is a constant, while each job completion time is a random variable due to machine breakdowns or other reasons of machine non-availability. In [5] (in [6,7], respectively), a flow-shop problem to stochastically minimize the makespan (the total completion time) was considered.

* Corresponding author.
E-mail address: sotskov@newman.bas-net.by (Yu.N. Sotskov).

0895-7177/$ – see front matter
doi:10.1016/j.mcm.2008.02.004

In this paper, we address another type of scheduling problem frequently encountered in realistic situations when it is hard to obtain a reliable probability distribution for each random processing time. In particular, we consider the following non-preemptive two-machine flow-shop problem with a scheduling objective to minimize the makespan. There are $n \geq 2$ jobs $\mathcal{J} = \{J_1, J_2, \dots, J_n\}$ to be processed by two machines $\mathcal{M} = \{M_1, M_2\}$ with the same machine route: (M_1, M_2). Each job $J_i \in \mathcal{J}$ has to be processed first by machine M_1 (without preemption), and then by machine M_2. All the n jobs are available for processing at time-point $t_0 = 0$. Each of the processing time p_{ij} of job $J_i \in \mathcal{J}$ by machine $M_j \in \mathcal{M}$ may take any real value between a given lower bound p_{ij}^L and upper bound p_{ij}^U. In such a case, there may not exist a single schedule that remains optimal for all possible realizations of the processing times. For a *solution* to this problem, we seek a minimal set of dominant schedules (such a solution concept was introduced in [8]).

Let $C_i(\pi)$ denote the completion time of job $J_i \in \mathcal{J}$ in the schedule π, and let $C_{\max}$ denote a minimization of the schedule length $C_{\max}(\pi)$: $C_{\max} = \min_{\pi \in S} C_{\max}(\pi) = \min_{\pi \in S}\{\max\{C_i(\pi) \mid J_i \in \mathcal{J}\}\}$, where S is the set of all feasible schedules containing at least one optimal schedule for the makespan criterion. By adopting the three-field notation introduced in [9], we denote the above problem as $F2|p_{ij}^L \leq p_{ij} \leq p_{ij}^U|C_{\max}$. We let T denote the set of all feasible vectors $p = (p_{1,1}, p_{1,2}, \dots, p_{n1}, p_{n2})$ of the uncertain processing times:

$$T = \{p \mid p_{ij}^L \leq p_{ij} \leq p_{ij}^U, J_i \in \mathcal{J}, M_j \in \mathcal{M}\}. \tag{1}$$

We note that the uncertainties of the processing times in problem $F2|p_{ij}^L \leq p_{ij} \leq p_{ij}^U|C_{\max}$ are due to external forces while in a scheduling problem with *controllable* processing times the objective is both to set the processing times and find an optimal schedule (see, e.g., articles [10–13]).

Our approach was originally proposed in [8] and developed in [14,15] for the $C_{\max}$ criterion, and in [16] for the total completion time criterion, $\sum C_i$. In particular, the formula for calculating the stability radius of an optimal schedule (i.e. the largest value of simultaneous independent variations of the job processing times for the schedule to remain optimal) has been provided in [8]. In the work of [16], the stability analysis of a schedule minimizing the total completion time was exploited in a branch and bound method for solving the job-shop problem $Jm|p_{ij}^L \leq p_{ij} \leq p_{ij}^U|\sum C_i$ with m machines. In [17], for a two-machine flow-shop problem $F2|p_{ij}^L \leq p_{ij} \leq p_{ij}^U|C_{\max}$ sufficient conditions have been identified when the transposition of two jobs minimizes the makespan. Article [18] addresses the total completion time in a flow-shop with the interval processing times. In particular, a geometrical algorithm has been developed for solving the flow-shop problem $Fm|p_{ij}^L \leq p_{ij} \leq p_{ij}^U, n = 2|\sum C_i$ with m machines and two jobs. For a flow-shop problem with two and three machines, sufficient conditions have been identified when the transposition of two jobs minimizes the total completion time. Work of [19] is devoted to the case of separate setup times with the criterion of minimizing the makespan or the total completion time. Namely, the processing times are fixed while each setup time is relaxed to be a distribution-free random variable within a given lower and upper bound. Local and global dominance relations have been identified for such a flow-shop problem with two machines. In [14,20], the necessary and sufficient conditions were proven for the case when a single 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 the interval processing times.

In this paper we will show how to use a minimal set of the dominant schedules obtained in the off-line scheduling phase (before the schedule execution) to best execute the schedule in the on-line phase by taking advantage of the on-line information on each uncertain processing time, where each uncertain processing of an operation will become realized at the completion of the operation. To demonstrate the strength of our approach, we also conduct extensive computational experiments for randomly generated problems $F2|p_{ij}^L \leq p_{ij} \leq p_{ij}^U|C_{\max}$ with n jobs from the range [10, 1000].

This paper is organized as follows. Section 2 is on definitions, notations and preliminary results. Section 3 provides an example to illustrate the main ideas used in the on-line scheduling phase. The definition of a dominant set of schedules is given in Section 4. Two cases will be considered in this paper on when the realized values of the uncertain processing times are available in the on-line scheduling phase. Sufficient conditions for schedule domination are proven in Sections 5 and 6, respectively, for the on-line scheduling phase corresponding to each of the two cases. Sufficient conditions for schedule domination in the off-line scheduling phase are proven in Section 7. Computational results for randomly generated instances are given in Section 8. We conclude with Section 9.

2. Preliminaries

Problem $Fm|p_{ij}^L \leq p_{ij} \leq p_{ij}^U|C_{\max}$ will be called the *uncertain* flow-shop problem while problem $Fm \parallel C_{\max}$ called the *deterministic* flow-shop problem.

If equality $p_{ij}^L = p_{ij}^U$ holds for each job $J_i \in \mathcal{J}$ and each machine $M_j \in \mathcal{M}$, then uncertain problem $F2|p_{ij}^L \leq p_{ij} \leq p_{ij}^U|C_{\max}$ reduces to deterministic flow-shop problem $F2 \parallel C_{\max}$, which is polynomially solvable due to Johnson [21]. Let $S = \{\pi_1, \pi_2, \dots, \pi_{n!}\}$ be the set of all permutations of the n jobs from set $\mathcal{J}$:

$$\pi_k = \{J_{k_1}, J_{k_2}, \dots, J_{k_n}\}, k \in \{1, 2, \dots, n!\}, \{k_1, k_2, \dots, k_n\} = \{1, 2, \dots, n\}.$$

Set S, with a cardinality of $|S| = n!$, defines the set of all *permutation schedules*. (In a *permutation schedule*, all jobs go through the machines from set $\mathcal{M}$ in the same order defined by this permutation.) From [21], it takes $O(n \log_2 n)$ time to construct a permutation $\pi_i = (J_{i_1}, J_{i_2}, \dots, J_{i_n}) \in S$ satisfying condition

$$\min\{p_{i_k 1}, p_{i_m 2}\} \leq \min\{p_{i_m 1}, p_{i_k 2}\} \tag{2}$$

for $1 \le k < m \le n$, and this permutation defines an optimal schedule to problem $F2 \parallel C_{\max}$. Algorithm for constructing a permutation $\pi_i \in S$ satisfying condition (2) (called Johnson's permutation) for the problem $F2 \parallel C_{\max}$ is based on the following rule.

Johnson's rule: Partition set $\mathcal{J}$ into two disjoint subsets N_1 and N_2 such that N_1 contains the jobs with $p_{i1} \le p_{i2}$ and N_2 the jobs with $p_{i1} \ge p_{i2}$. (The jobs with equality $p_{i1} = p_{i2}$ may be either in set N_1 or N_2.) In an optimal schedule, the jobs from set N_1 are processed first and are processed in non-decreasing order of p_{i1}. The jobs from set N_2 follow the jobs in N_1 in non-increasing order of p_{i2}. (Ties are broken arbitrarily.)

Remark 1. For the problem $F2||C_{\max}$ some optimal permutation schedules may not satisfy condition (2). In other words, inequalities (2) are *sufficient* for the optimality of permutation $\pi_i \in S$ but not *necessary* for the permutation optimality.

We note that the set S of all *permutation schedules* defined above is the *dominant* set of schedules for problem $F2 \parallel C_{\max}$ [21]. Since, for each fixed vector $p \in T$ of job processing times, since problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$ reduces problem $F2||C_{\max}$, it is sufficient to look for an optimal schedule among set S. Therefore, when solving uncertain problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$, it is sufficient to examine the set S.

Since preemption is not allowed in problem $F2||C_{\max}$, each permutation $\pi_k \in S$ defines a unique set of the earliest job completion times $C_1(\pi_k), C_2(\pi_k), \ldots, C_n(\pi_k)$ which in turn defines a unique *semiactive schedule*. (For a semiactive schedule, it is not possible to start any job earlier without starting another job later or without changing the order of the jobs on a machine.) In what follows, no distinction will be made between a permutation $\pi_k \in S$ and a semiactive schedule defined by this permutation. Such an agreement is needed for the uncertain problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$ since vector $p \in T$ of job processing times is unknown before scheduling. For the uncertain problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$ we use notation $C_j(\pi_k, p)$ to denote the completion time of job $J_j \in \mathcal{J}$, and notation $C_{\max}(\pi_k, p) = \max\{C_i(\pi_k, p) \mid J_i \in \mathcal{J}\}$ to denote the makespan for each fixed vector $p \in T$. The following definition, Definition 1, defines a *solution* to problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$.

Definition 1. Set of permutations $S(T) \subseteq S$ is defined to be a solution to problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$, if for any feasible vector $p \in T$ set $S(T)$ contains at least one Johnson's permutation for the problem $F2||C_{\max}$ associated with the vector p of job processing times provided that any proper subset of set $S(T)$ has no such a property.

From Definition 1 it follows that set $S(T)$ contains at least one optimal schedule $\pi_k \in S(T) \subseteq S$ for any given feasible vector $p \in T$ of job processing times: $C_{\max}(\pi_k, p) = \min\{C_{\max}(\pi_i, p) \mid \pi_i \in S\}$ and set $S(T)$ is a minimal set (with respect to inclusion) possessing such a property. We need to adopt such a dominant set $S(T)$ of permutations as a *solution* to the uncertain problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$ since for an uncertain scheduling problem a single dominant schedule typically does not exist.

In [15], the necessary and sufficient conditions have been identified for job $J_i \in \mathcal{J}$ to precede job $J_w \in \mathcal{J}$ in $S(T)$, i.e., there exists at least one Johnson's permutation of the form $\pi_k = (s_1, J_i, s_2, J_w, s_3) \in S$ for any feasible vector $p \in T$ of job processing times, where s_i means a subsequent (possibly, empty) of jobs from set $\mathcal{J}$. To facilitate our presentations of these conditions (and other results proven in Sections 4–6) we construct a partition $\mathcal{J} = \mathcal{J}_0 \cup \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}^*$ of the n jobs defined as follows:

$$\mathcal{J}_0 = \{J_i \in \mathcal{J} \mid p_{i1}^U \le p_{i2}^L, p_{i2}^U \le p_{i1}^L\}; \qquad \mathcal{J}^* = \{J_i \in \mathcal{J} \mid p_{i1}^U > p_{i2}^L, p_{i2}^U > p_{i1}^L\};$$
$$\mathcal{J}_1 = \{J_i \in \mathcal{J} \mid p_{i1}^U \le p_{i2}^L, p_{i2}^U > p_{i1}^L\} = \{J_i \in \mathcal{J} \setminus \mathcal{J}_0 \mid p_{i1}^U \le p_{i2}^L\};$$
$$\mathcal{J}_2 = \{J_i \in \mathcal{J} \mid p_{i1}^U > p_{i2}^L, p_{i2}^U \le p_{i1}^L\} = \{J_i \in \mathcal{J} \setminus \mathcal{J}_0 \mid p_{i2}^U \le p_{i1}^L\};$$

where $\mathcal{J}_0$, $\mathcal{J}^*$, $\mathcal{J}_1$, and $\mathcal{J}_2$ may be empty. Since for each job $J_k \in \mathcal{J}_0$, inequalities $p_{k1}^U \le p_{k2}^L$ and $p_{k2}^U \le p_{k1}^L$ imply equalities $p_{k1}^L = p_{k1}^U = p_{k2}^L = p_{k2}^U$, p_{k1} and p_{k2} are not only constants but equal: $p_{k1} = p_{k2} := p_k$. Sets $\mathcal{J}_1$ and $\mathcal{J}_2$ are defined in such a way that both inclusions $\mathcal{J}_1 \subseteq N_1$ and $\mathcal{J}_2 \subseteq N_2$ may hold for any vector $p \in T$ of job processing times (sets N_1 and N_2 are those used in Johnson's rule). The jobs in set $\mathcal{J}_0$ may be either in set N_1 or N_2 regardless of any realization of the vector $p \in T$ of job processing times. The jobs in set $\mathcal{J}^*$ may be either in set N_1 or N_2 depending on the realization of the vector $p \in T$ of job processing times. The following claim has been proven in [15].

Theorem 1 (*[15]*). *There exists a solution $S(T)$ to the problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$ with job $J_i \in \mathcal{J}$ preceding $J_w \in \mathcal{J}$ if and only if at least one of the following conditions holds:*

$$p_{w2}^U \le p_{w1}^L \quad and \quad p_{i1}^U \le p_{i2}^L; \tag{3}$$
$$p_{i1}^U \le p_{w1}^L \quad and \quad p_{i1}^U \le p_{i2}^L; \tag{4}$$
$$p_{w2}^U \le p_{w1}^L \quad and \quad p_{w2}^U \le p_{i2}^L. \tag{5}$$

Note that if condition (3) holds, then job J_i belongs to set N_1 and job J_w belongs to set N_2 for all feasible vectors $p \in T$ of job processing times. If condition (4) holds, then job J_i belongs to set N_1 for all vectors $p \in T$, while job J_w may be either in set N_1 or N_2 depending on the realizations of job processing times. If condition (5) holds, then job J_w belongs to set N_2 for all vectors $p \in T$, while job J_i may be either in set N_1 or N_2 depending on the realizations of the vectors $p \in T$.

Due to Theorem 1 if for a pair of jobs $J_i \in \mathcal{J}$ and $J_w \in \mathcal{J}$ at least one condition from (3) to (5) holds, then there exists a solution $S(T)$ to the uncertain problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$ with job J_i preceding J_w. Thus, via testing pairs of inequalities (3)–(5), one can construct a binary relation $\preceq$ (i.e., a subset of the Cartesian product $\mathcal{J} \times \mathcal{J}$) over the set $\mathcal{J}$ as follows: relation $J_i \preceq J_w$ holds if there exists a solution $S(T)$ to the problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$ such that job J_i precedes J_w in all permutations $\pi_k \in S(T)$. Using Theorem 1, to construct a binary relation $\preceq$ over the set $\mathcal{J}$ takes $O(n^2)$ time.

First, let us consider the case when $\mathcal{J}_0 = \emptyset$. For each pair of jobs $J_i \in \mathcal{J}_1$ and $J_w \in \mathcal{J}_1$ (or for each pair of jobs $J_i \in \mathcal{J}_2$ and $J_w \in \mathcal{J}_2$), there may exist a solution $S(T) \subset S$ with job J_i preceding J_w for all permutations $\pi_k \in S(T)$ or the other way around. In such a case, we can further define a strict precedence relation $\prec$: if $J_i \preceq J_w$ and $J_w \not\preceq J_i$, then $J_i \prec J_w$. If $J_i \preceq J_w$ and $J_w \preceq J_i$ with $i < w$, then $J_i \prec J_w$ and $J_w \not\prec J_i$. Since set $\mathcal{J}_0$ is empty, we obtain an antireflective, antisymmetric, and transitive binary relation $\prec$ over set $\mathcal{J} = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$, i.e., a strict order. Obviously, the strict order $\prec$ is uniquely defined for the problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$ with $\mathcal{J}_0 = \emptyset$.

Strict order $\prec$ may be represented by an acyclic digraph $G = (\mathcal{J}, A)$ with vertex set $\mathcal{J}$ or what is simpler by a reduction graph $G^* = (\mathcal{J}, A^*)$ of this strict order. Arc set A^* is defined as follows:

$$A^* = \{(J_i, J_w) \mid J_i \prec J_w \text{ and there is no job } J_k \text{ that } J_i \prec J_k \text{ and } J_k \prec J_w\}.$$

Reduction digraph $G^* = (\mathcal{J}, A^*)$ of strict order $\prec$ provides a compact representation for all permutations in solution $S^*(T)$.

Now, let us consider the case of $\mathcal{J}_0 \ne \emptyset$. The jobs J_k of set $\mathcal{J}_0 \ne \emptyset$ play a specific role in constructing a solution to the problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$: if $J_i \preceq J_k$ and $J_k \preceq J_i$, then there exist solutions $S_l(T)$ and $S_j(T)$ such that for each permutation of set $S_l(T)$ job J_k precedes J_i, while for each permutation of set $S_j(T)$ job J_k precedes J_i. Consequently, we can construct a family of solutions $\{S_j(T)\} = \{S_1(T), S_2(T), \dots, S_m(T)\}$ to the problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$ via fixing job $J_k \in \mathcal{J}_0$ at the candidate positions. Instead of using a single solution $S^*(T)$ as in the case of $\mathcal{J}_0 = \emptyset$, using a family of solutions $\{S_j(T)\}$ in the case of $\mathcal{J}_0 \ne \emptyset$ offers more flexibility in the on-line scheduling phase.

Remark 2. Let $J_i \preceq J_k, J_k \not\preceq J_i, J_k \preceq J_w$, and $J_w \not\preceq J_k$. In what follows, we will consider only a family of solutions $\{S_j(T)\}$ in which for each set $S_j(T)$ of the family $\{S_j(T)\}$, each job $J_k \in \mathcal{J}_0$ locates at some position between job $J_i \in \mathcal{J}_1 \cup \mathcal{J}^*$ and job $J_w \in \mathcal{J}_2 \cup \mathcal{J}^*$ for all permutations of set $S_j(T)$.

We will consider only the family of solutions $\{S_j(T)\}$ defined in Remark 2 since in Sections 5, 6 and 8 we will take advantage of the *local* information to schedule the conflicting jobs that compete for the same machine at the same time.

Based on Remark 2, for each job J_k of the set $\mathcal{J}_0 \ne \emptyset$, we can define the *candidate area* of job J_k for the solutions of the family $\{S_j(T)\}$ as follows. If $J_k \in \mathcal{J}_0$, then there exist jobs J_u and J_v such that the following equalities hold:

$$p_{u1}^L = \max\{p_{i1}^L \mid p_{i1}^L < p_k, J_i \in \mathcal{J}_1 \cup \mathcal{J}^*\}, \tag{6}$$

$$p_{v2}^L = \max\{p_{i2}^L \mid p_{i2}^L < p_k, J_i \in \mathcal{J}_2 \cup \mathcal{J}^*\}. \tag{7}$$

If job J_u locates at the r-th position and job J_v at the q-th position in a permutation $\pi_j \in S_j(T)$ $(r < q - 1)$, then job J_k may locate at any position between the r-th and the q-th position. The set of positions $r + 1, r + 2, \dots, q - 1$ between job J_u and J_v will be called the *candidate area* of job J_k. There are $q - r + 1$ positions in the candidate area of job J_k. It is clear that the following claim is correct.

Proposition 1. *Let $J_k \in \mathcal{J}_0, J_l \in \mathcal{J}_0$, and inequality $p_k \le p_l$ hold. Then the candidate area of job J_k in the family of solutions $\{S_j(T)\}$ to the problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$ contains the candidate area of job J_l.*

If $\mathcal{J}_0 \ne \emptyset$, then by using Theorem 1, one can construct a family of solutions $\{S_j(T)\}$ to the problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$ (instead of a unique solution $S_j^*(T)$ defined by digraph $G = (\mathcal{J}, A)$, if $\mathcal{J}_0 = \emptyset$). It is interesting to note that each job J_k of set $\mathcal{J}_0$ can serve as a buffer to absorb the uncertainties in the processing time of a job on a machine. To illustrate this idea, we consider in the next section an illustrative example with eleven jobs.

3. Illustrative example

We demonstrate how to best execute a schedule and possibly construct an actually optimal schedule for the problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$ with the intervals of the job processing times given in Table 1. We mean an actually optimal schedule in the sense that even though the processing times are uncertain a priori, the scheduler ends up with executing an optimal schedule as the scheduler has already known the realized values of all uncertain processing times beforehand.

There are two phases in the scheduling process: the off-line phase (the schedule planning phase) and the on-line phase (the schedule execution phase). The information of the lower and upper bounds for each uncertain processing time is available at the beginning of the off-line phase while the local information on the realization (the actual value) of each uncertain processing time is available once the corresponding operation (of a job on a machine) is completed. In the off-line phase, a family of solutions $\{S_j(T)\}$ is constructed first, which is useful in aiding a scheduler to best execute the schedule during the on-line phase.

For this example, subsets of set $\mathcal{J}$ in partition $\mathcal{J} = \mathcal{J}_0 \cup \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}^*$ are as follows:

$$\mathcal{J}_0 = \{J_1, J_4\}, \mathcal{J}^* = \{J_8, J_9, J_{10}\}, \qquad \mathcal{J}_1 = \{J_2, J_3, J_5, J_6, J_7\}, \mathcal{J}_2 = \{J_{11}\}.$$

Table 1
Intervals of the job processing times

i	1	2	3	4	5	6	7	8	9	10	11
p^L_{i1}	1	1	2	3	3	3	5	5	5	5	4
p^U_{i1}	1	1	2	3	4	5	5	11	7	8	4
p^L_{i2}	1	2	3	3	5	5	6	10	6	7	3
p^U_{i2}	1	3	3	3	8	8	6	11	7	9	3

Using Theorem 1, we obtain a partial strict order $\prec$ over the set $\mathcal{J} \setminus \mathcal{J}_0$ as follows:

$$(J_2 \prec J_3 \prec \{J_5, J_6\} \prec J_7 \prec \{J_8, J_9, J_{10}\} \prec J_{11}). \tag{8}$$

The partial sequence of (8) means that neither the order of jobs J_5 and J_6 is fixable nor the order of jobs J_8, J_9 and J_{10} is fixable for any solution $S_i(T)$ of the family $\{S_j(T)\}$. We now demonstrate how to best execute a schedule and possibly find an optimal schedule from set $S_i(T)$ of the family $\{S_j(T)\}$. Since the order of some jobs in set $S_i(T)$ is not fixable, there does not exist a dominant permutation that remains optimal for all feasible realizations of the job processing times. It is interesting that a scheduler may possibly find an actually optimal schedule by making a real-time scheduling decision at each decision-making time-point t_i of the completion time of job J_i on the first machine (machine M_1) as soon as the exact processing times are available for those operations completed before or at time-point $t_i > t_0 = 0$.

At time-point $t_0 = 0$, either job J_1 or job J_2 may be started on machine M_1 in an optimal way due to the above family of solutions $\{S_j(T)\}$. (As it will be clear later, it is better to process job $J_2 \in \mathcal{J}_1$ first.) Fig. 1 illustrates part of the scheduling process, where the candidate set $\{J_1, J_2\}$ of jobs for processing next is indicated at the top.

Let $c_1(i)$ and $c_2(i)$ denote completion time of job $J_i \in \mathcal{J}$ by machine M_1 and by machine M_2, respectively. We will consider the decision-making time-point $t_i = c_1(i)$ at which job J_i is completed on machine M_1 and a scheduler has to decide on the next job to be processed on machine M_1. In particular at time-point $t_2 = c_1(2) = 1$, machine M_1 completes the processing of job J_2 and machine M_2 will start to process this job. A scheduler now has to select a job from set $\{J_1, J_3\}$ for processing next on machine M_1. (Again, it will be clear later that it is better to select job $J_3 \in \mathcal{J}_1$ as the next job.) Thus, at time-point $t_2 = 1$, machine M_1 starts to process job J_3 for $p_{3,1} = 2$ time units.

At time-point $t_3 = c_1(3) = 3$, machine M_1 completes the processing of job J_3 and the candidate set for processing next on machine M_1 is $\{J_1, J_4, J_5, J_6\}$. At this time-point $t_3 = 3$, machine M_2 still is processing job J_2. The relations $p^U_{5,1} = 4 > p^L_{3,2} = 3$ and $p^U_{6,1} = 5 > p^L_{3,2} = 3$ hold for jobs J_5 and J_6, and the selection of job J_5 or job J_6 for processing next may cause idle time on machine M_2. In such a case, a scheduler can select job J_1 from set $\{J_1, J_4, J_5, J_6\}$ for processing immediately after job J_2. Such a selection of job $J_1 \in \mathcal{J}_0$ will allow a scheduler to delay the decision-making of sequencing jobs J_5 and J_6 until the time-point $t_1 = 3 + 1 = 4$ and thus to collect more realized values of the uncertain job processing times.

Let the realization (actual value) $p^*_{2,2}$ of the processing time $p_{2,2}$ of job J_2 turn out to be equal to $3 = p^*_{2,2}$. (Hereafter, we use notation p^*_{ij} for actual job processing time p_{ij}.) Then at time-point $t_1 = c_1(1) = 4$, machine M_2 finishes the processing of job J_2, and 4 time units are needed to complete the processing of both jobs J_3 and J_1 on machine M_2 (3 time units for processing job J_3 and 1 time unit for processing job J_1). The following inequalities hold: $p^U_{5,1} = 4 \leq 4$, $p^U_{6,1} = 5 > 4$. For job J_5 the relation $p^U_{5,1} + p^U_{6,1} = 4 + 5 \leq p^L_{5,2} + 4 = 5 + 4$ holds. Therefore, jobs J_5 and J_6 can be optimally processed with job J_5 preceding J_6 (since such an order causes no idle time on machine M_2). Then, machine M_1 will process job J_7 immediately after job J_6 (since job $J_4 \in \mathcal{J}_0$ can be used as a buffer to absorb the uncertainties in the processing times later when necessary).

At time-point $t_6 = c_1(6) = 13$, when machine M_1 completes the processing of job J_6, a scheduler already knows all job processing times completing before and at time-point $t_6 = 13$. Let the realized values be as follows: $p^*_{5,1} = 4$, $p^*_{6,1} = 5$, $p^*_{5,2} = 5$.

At time-point $t_7 = c_1(7) = 18$, a scheduler has a choice for the next job to be processed on machine M_1 among the jobs J_4, J_8, J_9 and J_{10}. At time-point t_7, machine M_2 already processed job J_6 for 5 time units, and a scheduler has no sufficient information to optimally select a job from set $\{J_8, J_9, J_{10}\}$ for processing next due to the fact that relations $p^U_{8,1} = 11 > p^L_{7,2} = 6$, $p^U_{9,1} = 7 > p^L_{7,2} = 6$, $p^U_{10,1} = 8 > p^L_{7,2} = 6$ hold (i.e., any such selection may cause idle time on machine M_2). Now it is time for a scheduler to select job $J_4 \in \mathcal{J}_0$ for processing immediately after job J_7 on machine M_1. The role of job $J_4 \in \mathcal{J}_0$ seems like a buffer to absorb the uncertainties of some uncertain job processing times.

At time-point $t_4 = c_1(4) = 18 + 3 = 21$, a scheduler has the choice for processing the next job among the jobs of J_8, J_9 and J_{10}. Assuming that $p^*_{6,2} = 8$, we know that J_6 is still under processing at time-point t_7 and is finished just at time-point t_4 on machine M_2. Hence, we obtain equalities $p^*_{4,2} + p^*_{7,2} = 3 + 6 = 9$ and therefore inequalities $p^U_{8,1} = 11 > 9$, $p^U_{9,1} = 7 < 9$, $p^U_{10,1} = 8 < 9$ hold. In case a scheduler selects job J_8 to be processed next, there will be idle time on machine M_2. Thus, a scheduler can select a job from set $\{J_9, J_{10}\}$ to be processed next. Let us check the following relation for an order of the three jobs (J_i, J_{i+1}, J_{i+2}):

$$p^U_{i1} + p^U_{i+1,1} + p^U_{i+2,1} \leq 8 + p^L_{i2} + p^L_{i+1,2}. \tag{9}$$

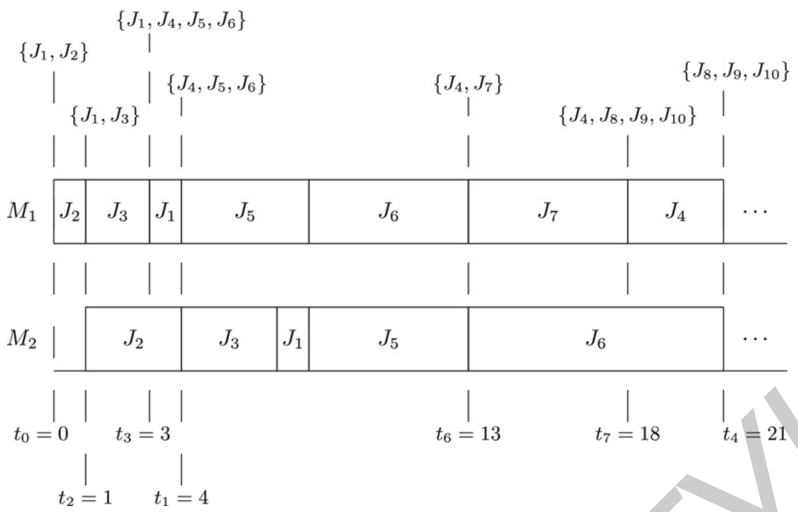

Fig. 1. Initial portion of the optimal schedule with processing times of jobs $\{J_1, J_2, \ldots, J_7\}$ given in (10).

The results of the checking for the four orders $\{(J_9, J_8, J_{10}), (J_9, J_{10}, J_8), (J_{10}, J_8, J_9), (J_{10}, J_9, J_8)\}$ are as follows:

$$p^U_{9,1} + p^U_{8,1} + p^U_{10,1} = 11 + 7 + 8 = 26 > 9 + p^L_{9,2} + p^L_{8,2} = 9 + 6 + 10 = 25,$$
$$p^U_{9,1} + p^U_{10,1} + p^U_{8,1} = 11 + 7 + 8 = 26 > 9 + p^L_{9,2} + p^L_{10,2} = 9 + 6 + 7 = 22,$$
$$p^U_{10,1} + p^U_{8,1} + p^U_{9,1} = 11 + 7 + 8 = 26 = 9 + p^L_{10,2} + p^L_{8,2} = 9 + 7 + 10 = 26,$$
$$p^U_{10,1} + p^U_{9,1} + p^U_{8,1} = 11 + 7 + 8 = 26 > 9 + p^L_{10,2} + p^L_{9,2} = 9 + 7 + 6 = 22.$$

Since relation (9) holds for the order (J_{10}, J_8, J_9), such an order will cause no idle time on machine M_2. Hence, a scheduler can optimally adopt the order (J_{10}, J_8, J_9) (since this order together with job J_{11} being the last one will be optimal for any feasible realization of the processing times of the remaining jobs $\{J_8, J_9, J_{10}, J_{11}\}$). Thus, we obtain the permutation: $\pi_u = (J_2, J_3, J_1, J_5, J_6, J_7, J_4, J_{10}, J_8, J_9, J_{11})$, which is necessarily optimal with the following partially realized values of job processing times (i.e., those for jobs $\{J_1, J_2, \ldots, J_7\}$):

$$p^*_{1,1} = 1, p^*_{1,2} = 1, p^*_{2,1} = 1, p^*_{2,2} = 3, p^*_{3,1} = 2, p^*_{3,2} = 3, p^*_{4,1} = 3,$$
$$p^*_{4,2} = 3, p^*_{5,1} = 4, p^*_{5,2} = 5, p^*_{6,1} = 5, p^*_{6,2} = 8, p^*_{7,1} = 5, p^*_{7,2} = 6. \quad (10)$$

The initial portion of this schedule is represented in Fig. 1. Note that the remaining portion of this schedule cannot be shown exactly since at time-point $t_4 = 21$ the processing times of jobs J_8, J_9, J_{10} and J_{11} are still unknown. But what is important, any feasible values of the remaining four jobs will not invalidate the optimality of permutation π_u. Thus, in spite of the job processing times being uncertain, a scheduler ends up with executing an actually optimal schedule from the family of sets $\{S_j(T)\}$.

The above two-phase scheduling process consists of the off-line planning phase with the family of sets $\{S_i(T)\}$ being constructed using Theorem 1, and the on-line execution phase with the following decision-making time-points: $t_2 = 1$, $t_3 = 3$, $t_1 = 4$, $t_6 = 13$, $t_7 = 18$ and $t_4 = 21$. The formal arguments of the above will be given in Sections 4–7.

4. Conditions for schedule domination

We first state the necessary and sufficient conditions for the existence of a single permutation $\pi_u \in S$ that remains optimal for all vectors $p \in T$ of job processing times, which have been proven in [20].

Theorem 2 *([20]). There exists a single-element solution $S(T) = \{\pi_u\} \subset S$ to the problem $F2|p^L_{ij} \le p_{ij} \le p^U_{ij}|C_{\max}$ if and only if*

(a) *for any pair of jobs J_i and J_j from set $\mathcal{J}_1$ (from set $\mathcal{J}_2$, respectively), either $p^U_{i1} \le p^L_{j1}$ or $p^U_{j1} \le p^L_{i1}$ (either $p^U_{i2} \le p^L_{j2}$ or $p^U_{j2} \le p^L_{i2}$),*

(b) *$|\mathcal{J}^*| \le 1$ and for job $J_{i^*} \in \mathcal{J}^*$ (if any), the following inequalities hold: $p^L_{i^*1} \ge \max\{p^U_{i1} \mid J_i \in \mathcal{J}_1\}$; $p^L_{i^*2} \ge \max\{p^U_{j2} \mid J_j \in \mathcal{J}_2\}$ and $\max\{p^L_{i^*1}, p^L_{i^*2}\} \ge p_k$ for each job $J_k \in \mathcal{J}_0$.*

We note that condition (a)–(b) is rarely satisfied in real situations. In Sections 5–7, we provide the sufficient conditions for an existence of a *dominant* set of permutations in the following sense.

Definition 2. Permutation $\pi_u \in S$ dominates permutation $\pi_k \in S$ with respect to T if inequality $C_{\max}(\pi_u, p) \le C_{\max}(\pi_k, p)$ holds for any vector $p \in T$ of job processing times. The set of permutations $S' \subseteq S$ is called dominant with respect to T if for each permutation $\pi_k \in S$ there exists permutation $\pi_u \in S'$ that dominates permutation π_k with respect to T.

If condition (a)–(b) of Theorem 2 holds, then singleton $\{\pi_u\}$ is dominant with respect to T (we say that such permutation π_u is dominant with respect to T). It is also clear that the set of permutations $S(T)$ used in Definition 1 is dominant with respect to T. It should noted that Definition 2 does not exploit Johnson's rule in contrast to Definition 1. In what follows, we will relax (if useful) the demand for a dominant permutation π_u to be a Johnson's one (see Remark 1).

In Sections 5–8, we will describe and justify the sufficient conditions and the formal algorithms for constructing a dominant permutation (if possible) for problem $F2|p^L_{ij} \le p_{ij} \le p^U_{ij}|C_{\max}$. Section 5 will consider the case of empty set $\mathcal{J}_0$. In Section 5, we will deal with first that there are two elements in solution $S^*(T)$, then that there are six elements in solution, and finally that the general case of $S^*(T)$. The case with non-empty set $\mathcal{J}_0$ will be utilized in Section 8. As far as the on-line scheduling phase is concerned, two cases will be distinguished:

(j) both the actual values p^*_{i1} and p^*_{i2} of job processing times p_{i1} and p_{i2} are available at time-point $t_i = c_1(i)$ when job J_i is completed by machine M_1;

(jj) the actual value p^*_{ij} of job processing time p_{ij} is available at time-point $t_i = c_j(i)$ when job J_i is completed by machine M_j.

Section 5 will address case (j), while Section 6 will address case (jj). We note that case (jj) is valid for almost all uncertain scheduling problems. Case (j) may occur in some real-world scheduling scenarios. One example is that M_1 is a diagnostic machine and M_2 is a repairing machine. During on-line scheduling phase, once a job J_i is completed on the diagnostic machine M_1, a scheduler usually knows the actual values (realized values) p^*_{i1} and p^*_{i2} of the processing times p_{i1} and p_{i2} of job J_i on both machines M_1 and M_2. Another example of case (j) is that machine M_1 is used for a *rough processing* of a part $J_i \in \mathcal{J}$ and machine M_2 is used for its *perfect processing*.

5. On-line scheduling in case (*J*)

5.1. Two conflicting jobs: $|S^*(T)| = 2$

Let $\mathcal{J}_0 = \emptyset$. Since there are only two permutations in $S^*(T)$, i.e., $S^*(T) = \{\pi_u, \pi_v\}$, it is clear that there exist only two non-adjacent vertices in the digraph $G = (\mathcal{J}, A)$ representing partial strict order $\prec$ defining solution $S^*(T) = \{\pi_u, \pi_v\}$. Due to Definition 1 permutation π_u (permutation π_v) is optimal Johnson's permutation for at least one feasible vector of job processing times but surely it is not Johnson's permutation for all feasible vectors $p \in T$ of job processing times (since condition (a)–(b) holds neither for permutation π_u nor for permutation π_v). W.l.o.g., we can assume that $\pi_u = (J_1, J_2, \ldots, J_{k-1}, J_k, J_{k+1}, \ldots, J_n)$ and $\pi_v = (J_1, J_2, \ldots, J_{k-1}, J_{k+1}, J_k, \ldots, J_n)$, i.e., only orders of jobs J_k and J_{k+1} are different in these two permutations. In what follows, if there is no path connecting vertex J_k with vertex J_{k+1} in the digraph $G = (\mathcal{J}, A)$, we say that jobs J_k and J_{k+1} are *conflicting*. Since order $(J_1, J_2, \ldots, J_{k-1})$ is the same in both permutations π_u and π_v defining solution $S^*(T) = \{\pi_u, \pi_v\}$, it is justified to process these jobs just in this order. Let the actual processing of jobs $J_1, J_2, \ldots, J_{k-1}$ be started (and completed) in order $J_1 \to J_2 \to \cdots \to J_{k-1}$ on both machines.

Since jobs J_k and J_{k+1} are conflicting, additional decision has to be used at time-point $t_{k-1} = c_1(k-1)$. It is clear that at time-point t_{k-1} the actual processing times of jobs from set $\mathcal{J}(t_{k-1}, 1) = \{J_1, J_2, \ldots, J_{k-1}\}$ on machine M_1 are already known. Let these actual values of processing times be as follows: $p_{1,1} = p^*_{1,1}, p_{2,1} = p^*_{2,1}, \ldots, p_{k-1,1} = p^*_{k-1,1}$. In case (j), the following assumption is made.

Assumption 1. The actual processing times of jobs from set $\mathcal{J}(t_{k-1}, 1)$ on machine M_2 are available at time-point $t_{k-1} = c_1(k-1)$: $p_{1,2} = p^*_{1,2}, p_{2,2} = p^*_{2,2}, \ldots, p_{k-1,2} = p^*_{k-1,2}$.

Thus at time-point $t_{k-1} = c_1(k-1)$, the following set of feasible vectors of processing times

$$T(k) = \{p \in T \mid p_{ij} = p^*_{ij}, J_i \in \mathcal{J}(t_{k-1}, 1), M_j \in \mathcal{M}\} \tag{11}$$

will be utilized instead of set T defined by equality (1). Next, we consider the following question. When will one of permutations π_u or π_v be optimal for all vectors $p \in T(k)$ of job processing times? To answer this question we have to consider all possible orders of the non-adjacent vertices J_k and J_{k+1} in the digraph $G = (\mathcal{J}, \mathcal{A})$ representing partial strict order $\prec$. (Due to Theorem 1 digraph G may be constructed in $O(n^2)$ time.)

At time-point t_{k-1} a scheduler has a choice between job J_k and J_{k+1} (which are conflicting) for processing next (immediately after job J_{k-1}) on machine M_1. Now a scheduler needs to test the condition in the following claim.

Proposition 2. *If condition*

$$c_2(k-1) - c_1(k-1) \ge p^U_{k1}, \tag{12}$$

$$c_2(k-1) - c_1(k-1) + p^L_{k2} \ge p^U_{k1} + p^U_{k+1,1} \tag{13}$$

holds, then permutation $\pi_u \in S^*(T) = \{\pi_u, \pi_v\}$ *is dominant with respect to* $T(k)$.

Proof. For permutation π_u and any vector $p \in T(k)$ of job processing times, we can calculate the earliest starting time $s_2(k+2)$ of job J_{k+2} on machine M_2 as follows: $s_2(k+2) = \max\{c_1(k+2), c_2(k+1)\}$. As we consider only semiactive schedules, machine M_1 processes all the jobs without any idle, so we obtain

$$c_1(k+2) = \sum_{i=1}^{k+2} p_{i1} = c_1(k-1) + p_{k1} + p_{k+1,1} + p_{k+2,1}.$$

For machine M_2, we obtain

$$c_2(k+1) = p_{k+1,2} + \max\{c_1(k-1) + p_{k1} + p_{k+1,1}, p_{k,2} + \max\{c_1(k-1) + p_{k1}, c_2(k-1)\}\}.$$

Inequality (12) implies equality $\max\{c_1(k-1)+p_{k1}, c_2(k-1)\} = c_2(k-1)$ for any vector $p \in T(k)$ of job processing times. Therefore we obtain

$$c_2(k+1) = p_{k+1,2} + \max\{c_1(k-1) + p_{k1} + p_{k+1,1}, p_{k,2} + c_2(k-1)\}. \tag{14}$$

Equality (14) means that machine M_2 has no idle time while processing jobs J_{k-1} and J_k.

Inequality (13) implies equality $\max\{c_1(k-1) + p_{k1} + p_{k+1,1}, p_{k,2} + c_2(k-1)\} = p_{k,2} + c_2(k-1)$. Therefore $c_2(k+1) = p_{k+1,2} + p_{k,2} + c_2(k-1)$. This means that machine M_2 has no idle time while processing jobs J_k and J_{k+1}.

We obtain $s_2(k+2) = \max\{c_1(k-1) + p_{k1} + p_{k+1,1} + p_{k+2,1}, p_{k+1,2} + p_{k,2} + c_2(k-1)\}$. Due to Assumption 1, values $c_1(k-1)$ and $c_2(k-1)$ are available at time-point $c_1(k-1)$. As machine M_1 has no idle time while processing jobs from set $\{J_1, J_2, \dots, J_{k+2}\}$, it is impossible to reduce value $c_1(k-1)+p_{k1}+p_{k+1,1}+p_{k+2,1}$. Analogously, as machine M_2 has no idle time while processing jobs from set $\{J_{k-1}, J_k, J_{k+1}\}$, it is impossible to reduce value $p_{k+1,2} + p_{k,2} + c_2(k-1)$ by alternative order of the jobs J_k and J_{k+1}. Therefore, permutation π_u dominates permutation π_v with respect to $T(k)$ (regardless of the exact value $s_2(k+2)$). Since $S^*(T) = \{\pi_u, \pi_v\}$, permutation π_u is dominant with respect to $T(k)$. ■

Thus, if condition (12) –(13) of Proposition 2 holds, then the order $J_k \to J_{k+1}$ of jobs J_k and J_{k+1} is the optimal order of these two jobs in the remaining part of the optimal permutation. Note that in the illustrative example of Section 3, Proposition 2 was implicitly used in sequencing the order of jobs J_5 and J_6 at time-point $t_1 = 4$.

Proposition 3. *If $c_2(k-1) - c_1(k-1) \ge p^U_{k1} + p^U_{k+1,1}$, then each permutation from set $S^*(T) = \{\pi_u, \pi_v\}$ is dominant with respect to $T(k)$.*

Proof. From condition $c_2(k-1) - c_1(k-1) \ge p^U_{k1} + p^U_{k+1,1}$ we obtain that both inequalities $c_2(k-1) - c_1(k-1) \ge p^U_{k1}$ and $c_2(k-1) - c_1(k-1) + p^L_{k2} \ge p^U_{k1} + p^U_{k+1,1}$ hold. Thus, condition (12)–(13) of Proposition 2 holds for permutation $\pi_u \in S^*(T)$. Hence, permutation π_u is dominant with respect to $T(k)$. On the other hand, condition $c_2(k-1) - c_1(k-1) \ge p^U_{k1} + p^U_{k+1,1}$ implies that both inequalities $c_2(k-1) - c_1(k-1) \ge p^U_{k+1,1}$ and $c_2(k-1) - c_1(k-1) + p^L_{k+1,2} \ge p^U_{k1} + p^U_{k+1,1}$ hold, i.e., appropriate condition of Proposition 2 holds for permutation $\pi_v \in S^*(T)$ with alternative order of jobs J_k and J_{k+1}. Hence, permutation π_v is dominant with respect to $T(k)$ as well. This completes the proof. ■

If condition of Proposition 3 holds, then the order of jobs J_k and J_{k+1} may be arbitrary in the remaining part of the optimal permutation. Similarly we can prove the following six sufficient conditions for domination of permutation π_u with respect to $T(k)$.

$$c_2(k-1) - c_1(k-1) < p^L_{k1}, \tag{15}$$

$$p^U_{k+1,1} \le p^L_{k2}, \tag{16}$$

$$p^L_{k+1,1} + p^L_{k+2,2} \ge p^U_{k2} + p^U_{k+1,1} \tag{17}$$

$$c_2(k-1) - c_1(k-1) + p^U_{k2} \le p^L_{k1} + p^L_{k+1,1}, \tag{18}$$

$$p^L_{k+1,1} \ge p^U_{k2}, \tag{19}$$

$$p^L_{k+2,1} \ge p^U_{k+1,2} \tag{20}$$

$$c_2(k-1) - c_1(k-1) \ge p^L_{k1}, \tag{21}$$

$$c_2(k-1) - c_1(k-1) < p^U_{k1}, \tag{22}$$

$$p^U_{k+1,1} \le p^L_{k2}, \tag{23}$$

$$p^L_{k+1,1} + p^L_{k+2,1} \ge p^U_{k2} + p^U_{k+1,2} \tag{24}$$

$$c_2(k-1) - c_1(k-1) \ge p^L_{k1}, \tag{25}$$

$$c_2(k-1) - c_1(k-1) < p^U_{k1}, \tag{26}$$

$$c_2(k-1) - c_1(k-1) + p^U_{k2} \ge p^L_{k1} + p^L_{k+1,1}, \tag{27}$$

$$p^U_{k+1,1} > p^L_{k2}, \quad (28)$$
$$p^L_{k+1,1} + p^L_{k+2,1} \geq p^U_{k2} + p^U_{k+1,2} \quad (29)$$
$$p^L_{k+2,1} \geq p^U_{k+1,2} \quad (30)$$
$$c_2(k-1) - c_1(k-1) \geq p^L_{k1}, \quad (31)$$
$$c_2(k-1) - c_1(k-1) < p^U_{k1}, \quad (32)$$
$$c_2(k-1) - c_1(k-1) + p^U_{k2} \geq p^L_{k1} + p^L_{k+1,1}, \quad (33)$$
$$p^U_{k+1,1} > p^L_{k2}, \quad (34)$$
$$p^U_{k+1,1} > p^L_{k2}, \quad (35)$$
$$p^L_{k+1,1} + p^L_{k+2,1} \geq p^U_{k2} + p^U_{k+1,2}, \quad (36)$$
$$p^L_{k+2,1} \geq p^U_{k+1,2} \quad (37)$$
$$c_2(k-1) - c_1(k-1) < p^L_{k1}, \quad (38)$$
$$p^U_{k2} \geq p^L_{k+1,1}, \quad (39)$$
$$p^U_{k+1,1} > p^L_{k2}, \quad (40)$$
$$p^L_{k+1,1} + p^L_{k+2,1} \geq p^U_{k2} + p^U_{k+1,2}. \quad (41)$$

The above sufficient conditions may be summarized in the following claim. (We omit its proof since it is similar to the proof of Proposition 2.)

Proposition 4. *If at least one from conditions (15)–(17), (18)–(20), (21)–(24), (25)–(30), (31)–(37) or (38)–(41) holds, then permutation $\pi_u \in S^*(T) = \{\pi_u, \pi_v\}$ is dominant with respect to $T(k)$.*

Thus, if at least one of the conditions of Propositions 2–4 holds, then a scheduler may fix the optimal order of jobs J_k and J_{k+1} regardless of the fact that the actual values of the processing times of the jobs $J_k, J_{k+1}, \ldots, J_n$ are still unavailable. In the next subsection, we show how to generalize Propositions 2–4 for the case when three jobs are conflicting at time-point $t_i > 0$.

5.2. Three conflicting jobs: $|S^*(T)| = 6$

Let jobs from set $\{J_k, J_{k+1}, J_{k+2}\} \subset \mathcal{J}$ be conflicting at time-point $t_{k-1} > 0$. So, there are six $(3! = 6)$ permutations in solution $S^*(T)$. We can test the following conditions similar to Propositions 2–4 and find a dominant permutation with respect to $T(k)$.

Proposition 5. *Let partial strict order $\prec$ over set $\mathcal{J} = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ be as follows $(J_1 \prec \cdots \prec J_{k-1} \prec \{J_k, J_{k+1}, J_{k+2}\} \prec J_{k+3} \prec \cdots \prec J_n)$. If $c_2(k-1) - c_1(k-1) > p^U_{k1}$, $c_2(k-1) - c_1(k-1) + p^L_{k2} > p^U_{k1} + p^U_{k+1,1}$ and $c_2(k-1) - c_1(k-1) + p^L_{k2} + p^L_{k+1,2} > p^U_{k1} + p^U_{k+1,1} + p^U_{k+2,1}$, then permutation $(J_1, \ldots, J_k, J_{k+1}, J_{k+2}, \ldots, J_n)$ is dominant with respect to $T(k)$.*

Proof. Arguing similarly as in the proof of Proposition 2, we conclude that machine M_1 has no idle time while processing jobs from set $\{J_1, J_2, \ldots, J_{k+3}\}$. Thus, it is impossible to reduce value $c_1(k+3)$ obtained for permutation $\pi_w = (J_1, \ldots, J_k, J_{k+1}, J_{k+2}, \ldots, J_n)$. Analogously, machine M_2 has no idle time while processing jobs $\{J_{k-1}, J_k, J_{k+1}, J_{k+2}\}$ in the order defined by permutation π_w. Thus, it is impossible to reduce value $c_2(k+2)$ defined for permutation π_w by alternative order of the jobs J_k, J_{k+1} and J_{k+2}. Therefore, if condition of Proposition 5 holds, then permutation $\pi_w \in S^*(T)$ is dominant with respect to $T(k)$ (regardless of the unknown value $s_2(k+3) = \max\{c_1(k+3), c_2(k+2)\}$). ■

If the condition of Proposition 5 holds, then in the remaining part of the optimal permutation, the order of jobs J_k, J_{k+1} and J_{k+2} is as follows: $J_k \to J_{k+1} \to J_{k+2}$. We can test the six propositions with conditions analogous to that of Proposition 5 but for different orders of three conflicting jobs. In the illustrative example of Section 3, Proposition 5 was implicitly used in the sequencing of the jobs J_8, J_9 and J_{10} at time-point $t_4 = 21$.

Similar to the proof of Proposition 3 we can prove the sufficient conditions for the existence of six dominant permutations as follows.

Proposition 6. *Let partial strict order $\prec$ over set $\mathcal{J} = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ be as follows $(J_1 \prec \cdots \prec J_{k-1} \prec \{J_k, J_{k+1}, J_{k+2}\} \prec J_{k+3} \prec \cdots \prec J_n)$. If $c_2(k-1) - c_1(k-1) > p^U_{k1} + p^U_{k+1,1} + p^U_{k+2,1}$, then each of six permutations from set $S^*(T)$ is dominant with respect to $T(k)$.*

If the condition of Proposition 6 holds, then the order of the three jobs J_k, J_{k+1} and J_{k+2} may be arbitrary in the remaining part of the optimal permutation.

Proposition 4 may be also generalized, and we can obtain the following fourteen sufficient conditions for an existence of a dominant permutation when $|S^*(T)| = 6$.

$$c_2(k-1) - c_1(k-1) < p^L_{k1}, \tag{42}$$
$$p^L_{k2} \geq p^U_{k+1,1}, \tag{43}$$
$$p^L_{k2} + p^L_{k+1,2} \geq p^U_{k+1,1} + p^U_{k+2,1}, \tag{44}$$
$$p^L_{k+1,1} + p^L_{k+2,1} + p^L_{k+3,1} \geq p^U_{k2} + p^U_{k+1,2} + p^U_{k+2,2}. \tag{45}$$
$$c_2(k-1) - c_1(k-1) + p^U_{k2} < p^L_{k1} + p^L_{k+1,1}, \tag{46}$$
$$p^L_{k+1,1} > p^U_{k2}, \tag{47}$$
$$p^U_{k+2,1} \leq p^L_{k+1,2}, \tag{48}$$
$$p^L_{k+2,1} + p^L_{k+3,1} \geq p^U_{k+1,2} + p^U_{k+2,2}. \tag{49}$$
$$c_2(k-1) - c_1(k-1) + p^U_{k2} + p^U_{k+1,2} < p^L_{k1} + p^L_{k+1,1} + p^L_{k+2,1}, \tag{50}$$
$$p^U_{k2} + p^U_{k+1,2} < p^L_{k+1,1} + p^L_{k+2,1}, \tag{51}$$
$$p^U_{k+1,2} < p^L_{k+2,1}, \tag{52}$$
$$p^L_{k+3,1} \geq p^U_{k+2,2}. \tag{53}$$
$$c_2(k-1) - c_1(k-1) < p^U_{k1}, \tag{54}$$
$$c_2(k-1) - c_1(k-1) \geq p^L_{k1}, \tag{55}$$
$$p^U_{k+1,1} \leq p^L_{k2}, \tag{56}$$
$$p^U_{k+1,1} + p^U_{k+2,1} \leq p^L_{k2} + p^L_{k+1,2}, \tag{57}$$
$$p^L_{k+1,1} + p^L_{k+2,1} + p^L_{k+3,1} \geq p^U_{k2} + p^U_{k+1,2} + p^U_{k+2,2}. \tag{58}$$
$$c_2(k-1) - c_1(k-1) + p^L_{k2} < p^U_{k1} + p^U_{k+1,1}, \tag{59}$$
$$c_2(k-1) - c_1(k-1) + p^U_{k2} \geq p^L_{k1} + p^L_{k+1,1}, \tag{60}$$
$$p^L_{k+1,1} > p^U_{k2}, \tag{61}$$
$$p^U_{k+2,1} \leq p^L_{k+1,2}, \tag{62}$$
$$p^L_{k+2,1} + p^L_{k+3,1} \geq p^U_{k+1,2} + p^U_{k+2,2}. \tag{63}$$
$$c_2(k-1) - c_1(k-1) + p^U_{k2} + p^U_{k+1,2} \geq p^L_{k1} + p^L_{k+1,1} + p^L_{k+2,1}, \tag{64}$$
$$c_2(k-1) - c_1(k-1) + p^L_{k2} + p^L_{k+1,2} < p^U_{k1} + p^U_{k+1,1} + p^U_{k+2,1}, \tag{65}$$
$$p^L_{k+1,1} + p^L_{k+2,1} > p^U_{k2} + p^U_{k+1,2}, \tag{66}$$
$$p^L_{k+2,1} > p^U_{k+1,2}, \tag{67}$$
$$p^L_{k+3,1} \geq p^U_{k+2,2}. \tag{68}$$
$$c_2(k-1) - c_1(k-1) < p^L_{k1}, \tag{69}$$
$$p^L_{k2} < p^U_{k+1,1}, \tag{70}$$
$$p^L_{k+1,1} \leq p^U_{k2}, \tag{71}$$
$$p^U_{k+2,1} \leq p^L_{k+1,2}, \tag{72}$$
$$p^L_{k+1,1} + p^L_{k+2,1} + p^L_{k+3,1} \geq p^U_{k2} + p^U_{k+1,2} + p^U_{k+2,2}. \tag{73}$$
$$c_2(k-1) - c_1(k-1) < p^L_{k1}, \tag{74}$$
$$p^U_{k2} + p^U_{k+1,2} \geq p^L_{k+1,1} + p^L_{k+2,1}, \tag{75}$$
$$p^L_{k+2,1} > p^U_{k+1,2}, \tag{76}$$

$$p^U_{k+1,1} + p^U_{k+2,1} > p^L_{k2} + p^L_{k+1,2}, \tag{77}$$
$$p^L_{k+1,1} + p^L_{k+2,1} + p^L_{k+3,1} \geq p^U_{k2} + p^U_{k+1,2} + p^U_{k+2,2}. \tag{78}$$
$$c_2(k-1) - c_1(k-1) + p^U_{k2} < p^L_{k1} + p^L_{k+1,1}, \tag{79}$$
$$p^U_{k+2,1} > p^L_{k+1,2}, \tag{80}$$
$$p^L_{k+2,1} \leq p^U_{k+1,2}, \tag{81}$$
$$p^L_{k+2,1} + p^L_{k+3,1} \geq p^U_{k+1,2} + p^U_{k+2,2}. \tag{82}$$
$$c_2(k-1) - c_1(k-1) \geq p^L_{k1}, \tag{83}$$
$$c_2(k-1) - c_1(k-1) < p^U_{k1}, \tag{84}$$
$$p^L_{k+1,1} \leq p^U_{k2}, \tag{85}$$
$$p^U_{k+1,1} > p^L_{k2}, \tag{86}$$
$$p^L_{k+1,2} \geq p^U_{k+2,1}, \tag{87}$$
$$p^L_{k+1,1} + p^L_{k+2,1} + p^L_{k+3,1} \geq p^U_{k2} + p^U_{k+1,2} + p^U_{k+2,2}. \tag{88}$$
$$c_2(k-1) - c_1(k-1) \geq p^L_{k1}, \tag{89}$$
$$c_2(k-1) - c_1(k-1) < p^U_{k1}, \tag{90}$$
$$p^U_{k+1,1} > p^L_{k2}, \tag{91}$$
$$p^L_{k+1,1} \leq p^U_{k2}, \tag{92}$$
$$p^L_{k+2,1} > p^U_{k+1,2}, \tag{93}$$
$$p^L_{k+3,1} \geq p^U_{k+2,2}, \tag{94}$$
$$p^L_{k+1,1} + p^L_{k+2,1} + p^L_{k+3,1} \geq p^U_{k2} + p^U_{k+1,2} + p^U_{k+2,2}. \tag{95}$$
$$c_2(k-1) - c_1(k-1) + p^L_{k2} < p^U_{k1} + p^U_{k+1,1}, \tag{96}$$
$$c_2(k-1) - c_1(k-1) + p^U_{k2} \geq p^L_{k1} + p^L_{k+1,1}, \tag{97}$$
$$p^U_{k+2,1} > p^L_{k+1,2}, \tag{98}$$
$$p^L_{k+2,1} \leq p^U_{k+1,2}, \tag{99}$$
$$p^L_{k+1,1} + p^L_{k+2,1} \geq p^U_{k2} + p^U_{k+1,2}, \tag{100}$$
$$p^L_{k+2,1} + p^L_{k+3,1} \geq p^U_{k+1,2} + p^U_{k+2,2}. \tag{101}$$
$$c_2(k-1) - c_1(k-1) < p^L_{k1}, \tag{102}$$
$$p^L_{k2} < p^U_{k+1,1}, \tag{103}$$
$$p^U_{k,1} \geq p^L_{k+1,1}, \tag{104}$$
$$p^U_{k+2,1} > p^L_{k+1,2}, \tag{105}$$
$$p^L_{k+2,1} \leq p^U_{k+1,2}, \tag{106}$$
$$p^L_{k+1,1} + p^L_{k+2,1} + p^L_{k+3,1} \geq p^U_{k2} + p^U_{k+1,2} + p^U_{k+2,2}. \tag{107}$$
$$c_2(k-1) - c_1(k-1) \geq p^L_{k1}, \tag{108}$$
$$c_2(k-1) - c_1(k-1) < p^U_{k1}, \tag{109}$$
$$p^L_{k+1,1} \leq p^U_{k2}, \tag{110}$$
$$p^U_{k+1,1} > p^L_{k2}, \tag{111}$$
$$p^L_{k+2,1} \leq p^U_{k+1,2}, \tag{112}$$
$$p^U_{k+2,1} > p^L_{k+1,2}, \tag{113}$$
$$p^L_{k+1,1} + p^L_{k+2,1} + p^L_{k+3,1} \geq p^U_{k2} + p^U_{k+1,2} + p^U_{k+2,2}. \tag{114}$$

The above sufficient conditions may be summarized in the following claim.

Proposition 7. *Let partial strict order $\prec$ over set $\mathcal{J} = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ be as follows $(J_1 \prec \cdots \prec J_{k-1} \prec \{J_k, J_{k+1}, J_{k+2}\} \prec J_{k+3} \prec \cdots \prec J_n)$. If at least one from conditions (42)–(45), (46)–(49), (50)–(53), (54)–(58), (59)–(63), (64)–(68), (69)–(73), (74)–(78), (79)–(82), (83)–(88), (89)–(95), (96)–(101), (102)–(107) or (108)–(114) holds, then permutation $\{J_1, \ldots, J_k, J_{k+1}, J_{k+2}, \ldots, J_n\}$ is dominant with respect to $T(k)$.*

Thus, if at least one from the sufficient conditions of Propositions 5–7 holds, then the order of jobs J_k, J_{k+1} and J_{k+2} must be $J_k \to J_{k+1} \to J_{k+2}$ in the remaining part of the optimal permutation. We can also test the above propositions with analogous conditions for other five possible orders of conflicting jobs J_k, J_{k+1} and J_{k+2}.

5.3. General case of solution $S^*(T)$

It is clear that Propositions 2–4 (Propositions 5–7, respectively) may be used if more than two (six) permutations are in the set $S^*(T)$ provided that, at each time-point of schedule execution, no more than two (three) jobs from set $\mathcal{J}$ are conflicting. We demonstrate this by the following example appropriate for using Propositions 2–4.

Let $|S^*(T)| = 8$ and six jobs from set $\mathcal{J} = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ be conflicting in a pairwise manner, e.g., a pair of jobs J_k and J_{k+1} are conflicting, a pair of jobs J_l and J_{l+1}, and a pair of jobs J_m and J_{m+1}. Then we can use Propositions 2–4 for each pair of jobs that are conflicting. W. l. o. g. we assume that the partial strict order $\prec$ over set $\mathcal{J} = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ is as follows: $(J_1 \prec \cdots \prec J_{k-1} \prec \{J_k, J_{k+1}\} \prec J_{k+2} \prec \cdots \prec J_{l-1} \prec \{J_l, J_{l+1}\} \prec J_{l+2} \prec \cdots \prec J_{m-1} \prec \{J_m, J_{m+1}\} \prec J_{m+2} \prec \cdots \prec J_n)$.

First, we process jobs $\{J_1, \ldots, J_{k-1}\}$ in the optimal order $J_1 \to \cdots \to J_{k-1}$. At time-point $t_{k-1} = c_1(k-1)$, we test Propositions 2–4 for pair of jobs $\{J_k, J_{k+1}\}$ that are conflicting. If at least one from conditions of Propositions 2–4 holds for order $J_k \to J_{k+1}$, then we process jobs $\{J_k, \ldots, J_{l-1}\}$ in the order $J_k \to \cdots \to J_{l-1}$. At time-point $t_{l-1} = c_1(l-1)$, we test Propositions 2–4 for pair of conflict jobs $\{J_l, J_{l+1}\}$. If at least one from conditions of Propositions 2–4 holds for order $J_l \to J_{l+1}$, then we process jobs $\{J_l, \ldots, J_{m-1}\}$ in the order $J_l \to \cdots \to J_{m-1}$. At time-point $t_{m-1} = c_1(m-1)$, we test Propositions 2–4 for pair of jobs $\{J_m, J_{m+1}\}$ that are conflicting. If at least one from conditions of Propositions 2–4 holds for order $J_m \to J_{m+1}$, then we process jobs $\{J_m, \ldots, J_n\}$ in the order $J_m \to \cdots \to J_n$.

Thus, if the condition of at least one of Propositions 2–4 holds for pairs of jobs $\{J_k, J_{k+1}\}$, $\{J_l, J_{l+1}\}$, and $\{J_m, J_{m+1}\}$, then we obtain dominant permutation $\pi_g = (J_1, \ldots, J_{k-1}, J_k, J_{k+1}, J_{k+2}, \ldots, J_{l-1}, J_l, J_{l+1}, J_{l+2}, \ldots, J_{m-1}, J_m, J_{m+1}, J_{m+2}, \ldots, J_n)$ with respect to $T(m)$ and so this permutation will be optimal for actual job processing times. Otherwise, e.g., if no condition of Propositions 2–4 holds for at least one pair of jobs $\{J_m, J_{m+1}\}$, then we obtain two-element dominant set of permutations $\{\pi_h, \pi_g\}$ where $\pi_g = (J_1, \ldots, J_{k-1}, J_k, J_{k+1}, J_{k+2}, \ldots, J_{l-1}, J_l, J_{l+1}, J_{l+2}, \ldots, J_{m-1}, J_{m+1}, J_m, J_{m+2}, \ldots, J_n)$ without proof that one of permutation π_h or π_g dominates another.

Furthermore, we can generalize the above sufficient conditions for the case when an arbitrary number of jobs are conflicting at the same on-line decision-making time-points. Let the set of r jobs be conflicting at time-point $t_k = c_1(k) > 0$. W. l. o. g. we assume that jobs from the set $\{J_{k_1}, J_{k_2}, \ldots, J_{k_r}\} \subset \mathcal{J} = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ are conflicting. Then we need test $r!$ possible orders of conflicting jobs. Generalization of Propositions 2 and 5 looks as follows.

Proposition 8. *Let partial strict order $\prec$ over set $\mathcal{J} = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ be as follows $(J_1 \prec \cdots \prec J_k \prec \{J_{k_1}, J_{k_2}, \ldots, J_{k_r}\} \prec J_{k+1} \prec \cdots \prec J_n)$. If inequality*

$$\sum_{i=1}^{s+1} p^L_{k_i 1} \le \sum_{j=0}^{s} p^U_{k_j 2}$$

holds for each $s = 0, 1, \ldots, r$, where $p^U_{k_0 2} = c_2(k) - c_1(k)$, then permutation $\{J_1, \ldots, J_k, J_{k_1}, J_{k_2}, \ldots, J_{k_r}, J_{k+1}, \ldots, J_n\}$ is dominant with respect to $T(k)$.

Generalization of Propositions 4 and 7 looks as follows.

Proposition 9. *Let partial strict order $\prec$ over set $\mathcal{J} = \mathcal{J}^* \cup \mathcal{J}_1 \cup \mathcal{J}_2$ be as follows $(J_1 \prec \cdots \prec J_k \prec \{J_{k_1}, J_{k_2}, \ldots, J_{k_r}\} \prec J_{k+1} \prec \cdots \prec J_n)$. If the following condition*

$$\sum_{i=m}^{s} p^L_{k_i 1} > \sum_{j=m-1}^{s-1} p^U_{k_j 2}, \quad m = 1, 2, \ldots, s,$$

$$\sum_{i=s+1}^{m} p^U_{k_i 1} \le \sum_{j=s}^{m-1} p^L_{k_j 2}, \quad m = s+1, s+2, \ldots, r, \qquad \sum_{i=s+1}^{r+1} p^L_{k_i 1} \ge \sum_{j=s}^{r} p^U_{k_j 2}$$

holds, where $p^U_{k_0 2} = c_2(k) - c_1(k)$, then permutation $\{J_1, \ldots, J_k, J_{k_1}, J_{k_2}, \ldots, J_{k_r}, J_{k+1}, \ldots, J_n\}$ is dominant with respect to $T(k)$.

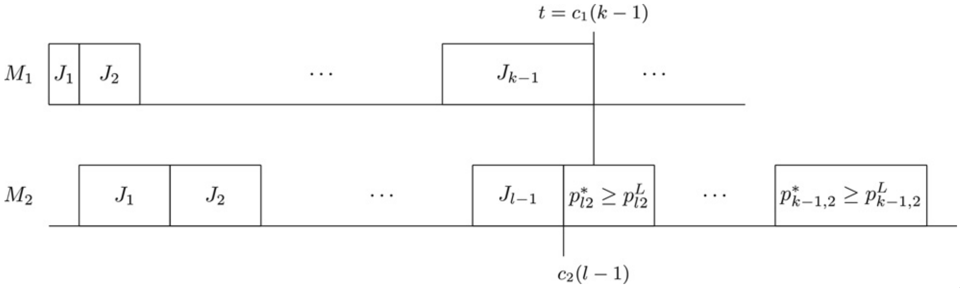

Fig. 2. Initial portion of the optimal schedule for jobs from set $\{J_1, J_2, \ldots, J_{k-1}\}$.

6. On-line scheduling in case (JJ)

In this section let us consider the case (jj). Now, the actual values p^*_{j2} of processing times p_{j2} of jobs J_j from set $\mathcal{J}(t_{l-1}, 2) = \{J_1, J_2, \ldots, J_{l-1}\}$ are available at time-point $t_{k-1} = c_1(k-1) > c_2(l-1)$, i.e., $p_{j2} = p^*_{j2}$, while the actual values of processing times p_{k2} of jobs J_k from set $\{J_l, J_{l+1}, \ldots, J_n\}$ are unavailable at time-point $t_{k-1} = c_1(k-1) < c_2(l)$. Thus, at time-point $t_{k-1} = c_1(k-1)$, the following set of feasible vectors

$$T(k, l) = \{p \in T \mid p_{i1} = p^*_{i1}, p_{j2} = p^*_{j2}, J_i \in \mathcal{J}(t_{k-1}, 1), J_j \in \mathcal{J}(t_{l-1}, 2)\}$$

of job processing times will be used instead of set $T(k)$ defined in (11).

Since Assumption 1 is not valid in case (jj), now we are forced to exploit the lower bounds $p^L_{l2}, p^L_{l2}, \ldots, p^L_{k-1,2}$ instead of the actual values $p^*_{l2}, p^*_{l2}, \ldots, p^*_{k-1,2}$ since the latter are unavailable at time-point $t_{k-1} = c_1(k-1)$. As a result we can calculate the lower bound $c^L_2(k-1)$ for the actual value $c_2(k-1)$ in the following way (see Fig. 2):

$$c^L_2(k-1) = c_2(l-1) + \max\{p^L_{l2}, c_1(k-1) - c_2(l-1)\} + \sum_{j=l}^{k-1} p^L_{j2}.$$

The analog of Proposition 2 is as follows.

Proposition 10. *If $c^L_2(k-1) - c_1(k-1) \geq p^U_{k1}$ and $c^L_2(k-1) - c_1(k-1) + p^L_{k2} \geq p^U_{k1} + p^U_{k+1,1}$, then permutation $\pi_u \in S^*(T) = \{\pi_u, \pi_v\}$ is dominant with respect to $T(k, l)$.*

We can calculate the following upper bound $c^U_2(k-1)$ for the actual value $c_2(k-1)$:

$$c^U_2(k-1) = c_2(l-1) + \sum_{j=l}^{k-1} p^U_{j2}.$$

Thus, the sufficient condition (15)–(17) from Proposition 4 can be reformulated as follows.

Proposition 11. *If $c^U_2(k-1) - c_1(k-1) < p^L_{k1}$, $p^U_{k+1,1} \leq p^L_{k2}$ and $p^L_{k+1,1} + p^L_{k+2,2} \geq p^U_{k2} + p^U_{k+1,1}$, then permutation $\pi_u \in S^*(T) = \{\pi_u, \pi_v\}$ is dominant with respect to $T(k, l)$.*

Propositions 5–9 can be reformulated for the case (jj) similarly.

7. Dominant permutation in off-line scheduling

In this section, we show that in the off-line scheduling phase, claims similar to Propositions 2–11 can also be applied along with Theorem 2. Recall that Theorem 2 provides the necessary and sufficient condition for an existence of Johnson's permutation that is dominant with respect to T. Due to a relaxation in requiring the permutation π_u to be a Johnson's one, we can obtain another sufficient conditions for an existence of a dominant permutation. To this end, it is necessary to substitute the exact difference $c_2(k-1) - c_1(k-1)$ (which is unavailable before time-point $t_0 = 0$) by its lower bound. It is clear that for the off-line scheduling phase there is no difference between case (j) and case (jj).

Let partial strict order $\prec$ defining solution $S^*(T)$ look as follows

$$(J_1 \prec \cdots \prec J_{k-1} \prec \{J_k, J_{k+1}\} \prec \cdots). \tag{115}$$

Then jobs J_k and J_{k+1} can be started on machine M_1 at time-point $t_{k-1} = c_1(k-1)$, and machine M_2 is available to process one of the jobs J_k or J_{k+1} from time-point $c_2(k-1)$. If at time-point $t \leq t_0 = 0$ a scheduler can calculate a lower bound Δ_{k-1} for the exact difference $c_2(k-1) - c_1(k-1)$, then before beginning the execution of a schedule a scheduler can test the conditions of Propositions 2–11 using value Δ_{k-1} instead of the difference $c_2(k-1) - c_1(k-1)$ unavailable at time-point t.

Next, we show how to calculate a tight lower bound Δ_{k-1}. If inclusion $J_i \in \mathcal{J}_1$ holds for $i = 1, 2, \ldots, k-1$, then for each index $i \in \{1, 2, \ldots, k-1\}$ the inequality $p_{i1}^U \le p_{i2}^L$ must hold and so $p_{i2} - p_{i1} \ge p_{i2}^L - p_{i1}^U \ge 0$. Thus, the following inequalities give a tight lower bound Δ_{k-1} for the difference $c_2(k-1) - c_1(k-1)$:

$$c_2(k-1) - c_1(k-1) \ge p_{11} + \sum_{i=1}^{k-1} p_{i2} - \sum_{i=1}^{k-1} p_{i1} = \sum_{i=1}^{k-1}(p_{i2} - p_{i1}) + p_{11} \ge \sum_{i=1}^{k-1}(p_{i2}^L - p_{i1}^U) + p_{11}^L = \Delta_{k-1}. \tag{116}$$

In the opposite case (if $J_i \notin \mathcal{J}_1$), a lower bound Δ_{k-1} for the difference $c_2(k-1) - c_1(k-1)$ may be calculated recursively as follows. If $|\mathcal{J}_1| = m$, we obtain

$$\Delta_m = \sum_{i=1}^{m}(p_{i2}^L - p_{i1}^U) + p_{11}^L$$

due to the last equality in (116) with $k-1 = m$. Further, for each index $l \in \{m+1, m+1, \ldots, k-1\}$ one can use the following recursive formula $\Delta_l = \max\{0,\ \Delta_{l-1} - p_{l,1}^U\} + p_{l,2}^L$. As a result we obtain the following claim similar to Proposition 2.

Proposition 12. *If $\Delta_{k-1} \ge p_{k1}^U$ and $\Delta_{k-1} + p_{k2}^L \ge p_{k1}^U + p_{k+1,1}^U$, then permutation $\pi_u \in S^*(T) = \{\pi_u, \pi_v\}$ is dominant with respect to T.*

Furthermore, all the propositions presented in Sections 5 and 6 can be reformulated for the case of off-line scheduling provided that the exact difference $c_2(k-1) - c_1(k-1)$ is substituted by the lower bound Δ_{k-1}. Note that Propositions 2–10 may be only used if $k > 1$ in the partial strict order (115). Let $k = 1$ and jobs J_1 and J_2 be conflicting, i.e., partial strict order (115) be as follows $(\{J_1, J_2\} \prec J_3 \prec \cdots)$. We will try to sequence two conflicting jobs in an optimal way before time-point $t_0 = 0$. Let us consider the case when machine M_2 has an idle time before processing job J_3. In this case, machine M_2 can process job J_3 from the time when machine M_1 completes the processing of this job (i.e., from time-point $t_3 = c_1(3)$). It is easy to prove the following sufficient condition.

Proposition 13. *If $p_{3,1}^L \ge p_{2,2}^U + \max\{0, p_{1,2}^U - p_{2,1}^L\}$, then the order of jobs J_1 and J_2 in the optimal permutation is $J_1 \to J_2$.*

Obviously, $c_2(2) - c_1(2) \le p_{2,2}^U + \max\{0, p_{1,2}^U - p_{2,1}^L\}$. In the latter inequality, the difference $p_{1,2}^U - p_{2,1}^L$ is equal to the maximal addition for the case where machine M_2 cannot finish job J_1 before machine M_1 has finished job J_2. Hence, we obtain inequality $c_1(3) > c_2(2)$, and machine M_2 has an idle time before processing job J_3. Therefore, in the opposite case (where the optimal order of jobs J_1 and J_2 cannot be defined by Proposition 13), we cannot decrease value $C_{\max}$. Of course, if $p_{3,1}^L > p_{2,2}^U + p_{1,2}^U$, then both permutations $\pi_u = (J_1, J_2, J_3, \ldots)$ and $\pi_v = (J_2, J_1, J_3, \ldots)$ are dominant and the optimal order of jobs J_1 and J_2 may be arbitrary. More precisely, if $p_{3,1}^L > \max\{p_{1,2}^U + p_{2,2}^U - \min\{p_{1,1}^L, p_{1,2}^L\}, \max\{p_{1,2}^U; p_{2,2}^U\}\}$, then both permutations π_u and π_v are dominant. In other words, if $p_{3,1}^L > \max\{0, p_{1,2}^U - p_{2,1}^L, p_{2,2}^U - p_{1,1}^L\} + \max\{p_{1,2}^U, p_{2,2}^U\}$, then both permutations π_u and π_v are dominant (arbitrary order of jobs J_1 and J_2 is optimal).

It is easy to see that the above propositions can be generalized for the case when more than two jobs are conflicting at time-point $t_0 = 0$. Next, we demonstrate such a generalization with two examples. E.g., for three conflicting jobs, we obtain the following claim.

Proposition 14. *Let partial strict order $\prec$ look as $(\{J_1, J_2, J_3\} \prec J_4 \prec \cdots)$. If $p_{4,1}^L \ge p_{3,2}^U + \max\{0,\ p_{2,2}^U - p_{3,1}^L + \max\{0, p_{1,2}^U - p_{2,1}^L\}\}$, then the order of jobs J_1, J_2 and J_3 in the optimal permutation is $J_1 \to J_2 \to J_3$.*

Let δ_m be defined recursively as follows: $\delta_m = \max\{0, p_{m,2}^U - p_{m+1,1}^L + \delta_{m-1}\}$, where $\delta_1 = \max\{0, p_{1,2}^U - p_{2,1}^L\}$. Using this notation, we can generalize the above Propositions 13 and 14 for the case of r conflicting jobs at time-point $t_0 = 0$.

Proposition 15. *Let partial strict order $\prec$ look as $(\{J_1, J_2, \ldots, J_r\} \prec J_{r+1} \prec \cdots)$. If $p_{r+1,1}^L \ge p_{r,2}^U + \delta_{r-1}$, then the order of jobs J_1, $J_2, \ldots, J_r$ in the optimal permutation is $J_1 \to J_2 \to \cdots \to J_r$.*

8. Algorithms and computational results

Our computational study of the two-phase scheduling was performed on a large number of randomly generated problems $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$. The following algorithms were coded in C+: Algorithm 1 for the off-line scheduling and Algorithm 2 (Algorigthm 3, respectively) for the on-line scheduling provided that set $\mathcal{J}_0$ is empty (nonempty).

Algorithm 1 (*For Off-Line Scheduling*).

Input: lower and upper bounds p_{ij}^L and p_{ij}^U for processing times p_{ij}
of jobs $J_i \in \mathcal{J}$ on machines $M_j \in \mathcal{M}$.
Output: solution $S(T)$ to the problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$;
binary relation $\preceq$ defining solution $S(T)$, if $|S(T)| > 1$.
*Step*1: test condition (a)–(b) of Theorem 2.
*Step*2: **IF** condition (a)–(b) holds **GOTO** step 8.

Table 2
Percentage of solved instances with empty set $\mathcal{J}_0$

n	L (%)	Number of decision points	Percentage of proved decisions	Off-line optimal (%)	On-line optimal (%)	Optimal without proof (%)	Max error of $C_{\max}$ (%)	Average error of $C_{\max}$ (%)
	1	19	73.68	82	13	3	6.230213	0.092163
	2	39	69.23	65	24	7	14.894381	0.302439
	3	54	64.81	52	31	15	7.330572	0.129230
	4	71	73.24	41	43	10	11.425118	0.186663
10	5	71	71.83	42	39	14	18.172153	0.332733
	6	93	70.97	27	49	18	12.438417	0.275810
	7	100	74.00	29	47	15	17.770338	0.274215
	8	121	66.94	19	51	21	24.294342	0.379896
	9	125	56.80	10	44	32	16.657515	0.952808
	10	130	67.69	16	50	24	18.044373	0.673897
	1	71	85.92	47	43	9	1.857141	0.018571
	2	126	91.27	17	72	9	14.779399	0.163159
	3	155	87.74	14	70	15	0.022465	0.000225
	4	211	90.05	4	76	18	7.927369	0.079337
20	5	238	84.87	1	70	22	10.647840	0.370658
	6	237	81.43	1	65	25	7.414827	0.277665
	7	288	84.03	0	66	28	7.479012	0.114603
	8	250	78.80	0	64	32	12.671661	0.314745
	9	294	82.31	0	60	25	10.750363	0.537575
	10	303	81.85	0	59	28	8.804494	0.366674
	1	134	97.76	26	71	3	0.000000	0.000000
	2	241	89.63	10	71	18	0.004085	0.000041
	3	319	93.73	0	83	13	3.760090	0.067121
	4	347	95.10	0	86	13	1.603097	0.016031
30	5	413	94.43	0	80	19	11.283378	0.112834
	6	390	88.46	0	69	23	6.422380	0.222811
	7	448	90.18	0	70	27	10.415929	0.198046
	8	450	90.67	0	69	26	0.192515	0.002738
	9	440	87.73	0	64	25	15.253723	0.399358
	10	446	89.01	0	67	26	11.338615	0.119900
	1	236	96.19	4	88	8	0.000000	0.000000
	2	421	94.54	0	84	14	0.269543	0.002735
40	3	484	96.69	0	88	11	9.348538	0.093485
	4	559	93.74	0	72	23	6.632380	0.101148
	5	581	95.87	0	83	16	1.854262	0.018543
	10	526	90.68	0	68	30	7.492048	0.074967
50	5	764	94.24	0	74	23	4.768900	0.047914
	10	616	89.29	0	60	31	3.341783	0.125370
60	5	889	93.70	0	63	32	8.621157	0.136614
	10	704	92.33	0	64	29	8.556119	0.250279
70	5	954	96.02	0	76	21	1.219268	0.012251
	10	716	92.18	0	63	31	14.920141	0.230028
80	5	1086	95.67	0	68	32	0.000000	0.000000
	10	784	91.20	0	63	30	6.311226	0.078879
90	5	1201	95.50	0	66	29	3.027027	0.048897
	10	776	90.08	0	51	37	15.321626	0.302109
100	5	1259	96.43	0	74	25	0.001865	0.000019
	10	750	89.73	0	50	37	5.903523	0.169706

*Step*3: using Theorem 1 construct digraph $G = (\mathcal{J}', A)$ with vertex set $\mathcal{J}' = \mathcal{J} \setminus \mathcal{J}_0$.
*Step*4: construct binary relation $\preceq$ by adding set $\mathcal{J}_0$ to digraph $G = (\mathcal{J}', A)$.
*Step*5: test conditions of Propositions 12–15.
*Step*6: **IF** there are no conflicting jobs **GOTO** step 8.
*Step*7: **STOP**
*Step*8: **STOP:** there exists dominant permutation with respect to T.

In Algorithm 2, integer k, $1 \leq k \leq n$, denotes the number of decision-making time-points $t_i = c_1(i), J_i \in \mathcal{J}$, in on-line scheduling phase. Integer m, $1 \leq m \leq k$, denotes the number of decision-making time-points for which the optimal orders of the conflicting jobs were found using the sufficient conditions from Propositions 2–10.

Table 3
Percentage of solved instances with 10% of jobs from set $\mathcal{J}_0$

n	L (%)	Number of decision points	Percentage of proved decisions	Off-line optimal (%)	On-line optimal (%)	Optimal without proof (%)	Max error of $C_{\max}$ (%)	Average error of $C_{\max}$ (%)
	1	18	72.22	79	18	2	17.966948	0.179669
	2	26	61.54	65	25	1	21.889175	1.026670
	3	49	69.39	49	37	8	12.544724	0.411497
	4	50	68.00	49	35	7	17.959299	0.656619
10	5	54	55.56	49	30	10	14.644439	0.911094
	6	73	64.38	26	49	14	14.280224	0.763463
	7	82	57.32	20	47	18	23.059456	1.461247
	8	92	59.78	21	46	17	16.038074	1.009285
	9	96	57.29	20	45	15	17.468323	1.431748
	10	114	56.14	10	49	19	19.623438	1.667662
	1	52	88.46	49	46	4	5.821766	0.058218
	2	99	92.93	18	75	3	6.804946	0.165536
	3	150	86.67	5	78	11	7.669424	0.257041
	4	154	85.71	5	77	6	8.306249	0.544108
20	5	190	85.26	6	71	12	15.721927	0.588305
	6	202	89.60	0	83	6	8.690571	0.590736
	7	233	86.70	1	75	13	9.809823	0.518078
	8	269	81.78	0	71	16	19.979408	0.823335
	9	247	79.76	0	64	10	10.719061	1.378448
	10	286	83.92	0	68	14	11.104570	1.122154
	1	133	93.98	14	80	5	3.743102	0.037431
	2	214	92.06	7	81	7	9.082943	0.171559
	3	275	93.82	2	84	11	6.714648	0.129161
	4	328	91.46	1	76	12	10.427673	0.408122
30	5	346	92.20	0	78	10	4.925068	0.523251
	6	340	88.53	0	68	15	13.092766	0.667315
	7	365	89.59	0	71	12	20.051733	0.891334
	8	424	90.80	0	71	10	13.626771	0.935853
	9	388	88.14	0	70	12	16.552177	0.836974
	10	416	88.70	0	68	19	9.795372	0.620623
	1	202	94.06	5	83	10	4.489825	0.085888
	2	354	94.35	0	85	13	6.027034	0.093683
40	3	428	95.33	0	87	8	4.454952	0.138229
	4	475	93.26	0	77	9	14.140268	0.689480
	5	513	96.69	0	88	6	17.602324	0.317764
	10	519	90.17	0	62	15	6.950419	0.735055
50	5	652	96.63	0	83	11	3.451158	0.161636
	10	580	89.83	0	58	12	6.377758	0.672041
60	5	739	95.81	0	75	12	3.175847	0.337660
	10	634	91.01	0	62	7	9.123387	0.838601
70	5	933	95.71	0	73	11	2.618088	0.303773
	10	729	93.14	0	66	13	8.430060	0.540960
80	5	992	94.76	0	72	8	9.327187	0.501414
	10	735	90.88	0	62	6	8.642805	0.703802
90	5	1113	96.14	0	76	10	11.927623	0.440951
	10	761	93.04	0	66	9	14.317971	0.661120
100	5	1083	95.57	0	68	7	4.340650	0.430289
	10	829	93.97	0	64	4	3.391186	0.517160

Algorithm 2 (*For On-Line Scheduling* ($\mathcal{J}_0 = \emptyset$)).

Input: lower and upper bounds p_{ij}^L and p_{ij}^U for processing times p_{ij}
of jobs $J_i \in \mathcal{J}$ on machines $M_j \in \mathcal{M}$;
solution $S^*(T)$, $|S^*(T)| > 1$, to the problem $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$
defined by partial strict order $\prec$.

Output: either dominant permutation $\pi_u \in S^*(T)$ with respect to $T(i)$, $J_i \in \mathcal{J}$,
or permutation from solution $S^*(T)$ without its optimality proof.

*Step*1: set $k := 0$, $m := 0$.

*Step*2: **IF** first jobs in partial strict order $\prec$ are conflicting **THEN**
BEGIN
check conditions of Proposition 15 for all orders of conflicting jobs

IF conditions of Proposition 15 holds for at least one order of conflicting jobs
THEN $k := k + 1$, $m := m + 1$, choose optimal order of conflicting jobs
ELSE $k := k + 1$, choose arbitrary order of conflicting jobs
and process conflicting jobs in the chosen order.
END

*Step*3: **UNTIL** finishing the last job in the actual schedule.
*Step*4: process linear part of partial strict order $\prec$.
*Step*5: check conditions of Propositions 8 and 9 for all orders of conflicting jobs.
IF there are two conflict jobs **THEN** check conditions of Proposition 4.
*Step*6: **IF** at least one sufficient condition from Propositions 2–11 holds
for at least one order of conflicting jobs
THEN $k := k + 1$, $m := m + 1$, choose optimal order of conflicting jobs,
ELSE $k := k + 1$, choose arbitrary order of conflicting jobs,
*Step*7: process conflicting jobs in the chosen order.
*Step*8: **RETURN**
*Step*9: **IF** $k = m$ **THEN GOTO** step 14.
*Step*10: calculate length $C_{\max}$ of the schedule that was constructed via steps 1 – 9 and
length $C^*_{\max}$ of the optimal schedule constructed for actual processing times.
*Step*11: **IF** $C_{\max} = C^*_{\max}$ **THEN GOTO** step 13.
*Step*12: **STOP:** constructed schedule is not optimal for actual processing times.
*Step*13: **STOP:** optimality of actual permutation is defined after schedule execution.
*Step*14: **STOP:** optimality of actual permutation is proven before schedule execution.

If $\mathcal{J}_0 \neq \emptyset$, then Algorithm 3 has to be used instead of Algorithm 2 at on-line scheduling phase. The former differs from the latter by the following part which has to be used instead of the above Steps 5–7. Moreover in Algorigthm 3, set $S^*(T)$ will be substituted by $S(T)$. Let N_j denote subset of set $\mathcal{J}_0$ of the jobs that can be processed at time-point $t_j = c_1(j)$.

Part of Algorithm 3 (*For On-Line Scheduling*).

*Step*5: check conditions of Propositions 8 and 9 for all orders of conflicting jobs.
IF there are only two conflict jobs at time point t_i
THEN check conditions of Proposition 4.
*Step*5a: **IF** no sufficient condition holds **THEN** calculate the corresponding
subset N_j.
*Step*5b: **UNTIL** $N_j = \emptyset$ **OR** at least one sufficient condition from
Propositions 2–11 holds for at least one order of conflicting jobs,
*Step*5c: process job $J_i \in N_j$ with the largest p_i and delete job J_i from set N_j.
*Step*6: **IF** at least one sufficient condition holds for at least one order
of conflicting jobs
THEN $k := k + 1$, $m := m + 1$, choose optimal order of conflicting jobs,
ELSE $k := k + 1$, choose arbitrary order of conflicting jobs.
*Step*7: process conflicting jobs in the chosen order.
*Step*7a: **RETURN**

For the computational experiments, we used a Celeron 1200 MHz processor with 384 MB main memory. We made 100 tests in each series, i.e. for each combination of n and L where L defines the percentage of relative error of input data (job processing times) known before scheduling. The lower bound p^L_{ij} and upper bound p^U_{ij} of job processing times are uniformly distributed in the range [10, 1000] in such a way that the following equality holds:

$$L = ((p^U_{ij} - p^L_{ij}) : (p^U_{ij} + p^L_{ij})/2) \cdot 100.$$

The bounds p^L_{ij} and p^U_{ij} and the actual processing times p_{ij} were decimal fractions with two digits after decimal point. Generater from [22] has been used for (pseudo) random generating instances of the problem $F2|p^L_{ij} \leq p_{ij} \leq p^U_{ij}|C_{\max}$.

It is easy to see that all sufficient conditions proven in Sections 5–7 may be tested in polynomial time of the number n of jobs. Moreover, to minimize running time of the Algorithms 1–3 these sufficient conditions were tested in an increasing order of their complexity up to the first positive answer (if any) to the following question. Does a dominant permutation exist at time-point $t_i = c_1(i)$?

In the experiments, the CPU-time was small: even for 1000 jobs both Algorithms 1 and 2 (Algorigthm 3) take less than 0.05 s for solving one instance of the problem $F2|p^L_{ij} \leq p_{ij} \leq p^U_{ij}|C_{\max}$. Therefore, we do not present the CPU-time in Tables 2–6.

Note that the results presented in Tables 2–6 have been only obtained for the case (jj) of on-line scheduling, i.e., Assumption 1 was not used and so the actual value of processing time p_{ij} became only known at time point $t_i = c_j(i)$ when job J_i was completed by machine M_j.

Table 4
Percentage of solved instances with 30% of jobs from set $\mathcal{J}_0$

n	L (%)	Number of decision points	Percentage of proved decisions	Off-line optimal (%)	On-line optimal (%)	Optimal without proof (%)	Max error of $C_{\max}$ (%)	Average error of $C_{\max}$ (%)
	1	9	66.67	85	12	0	12.769878	0.191195
	2	14	78.57	70	27	0	13.634037	0.347974
	3	31	61.29	59	31	2	13.529054	0.735699
	4	40	32.50	44	37	5	23.480253	1.282487
10	5	43	37.21	42	35	6	21.291142	1.924775
	6	47	53.19	36	42	8	17.709812	1.613568
	7	62	48.39	27	44	7	18.857832	2.174505
	8	54	46.30	38	35	5	26.330475	2.890409
	9	71	42.25	24	40	9	26.084694	2.628773
	10	67	50.75	26	47	5	21.843046	2.253550
	1	26	80.77	54	41	2	9.495034	0.189950
	2	66	72.73	25	59	6	9.860765	0.496118
	3	92	78.26	19	65	7	8.274606	0.526225
	4	126	79.37	8	73	10	8.872502	0.651189
20	5	123	82.11	4	76	6	9.250347	0.918365
	6	145	83.45	1	77	5	8.582347	1.017275
	7	172	74.42	1	67	9	13.947658	1.272475
	8	169	78.11	0	72	8	22.539068	1.723347
	9	203	78.33	0	73	12	10.772242	1.085768
	10	190	81.05	0	73	5	11.410321	1.304585
	1	74	97.30	26	72	1	5.184722	0.051847
	2	132	92.42	6	84	4	5.993108	0.310904
	3	195	92.31	1	87	8	5.672687	0.189419
	4	223	89.24	0	78	10	7.023677	0.547064
30	5	260	91.54	0	81	3	7.348124	0.853035
	6	274	87.96	0	75	5	8.557904	0.987270
	7	281	92.53	0	83	8	13.237219	0.531717
	8	310	87.42	0	71	10	6.351249	0.769261
	9	320	81.88	0	67	10	14.983966	1.227586
	10	330	85.45	0	62	14	15.381570	1.045122
	1	133	93.23	7	86	2	4.371631	0.119839
	2	209	91.87	0	84	11	4.267471	0.200966
40	3	264	93.94	0	87	8	6.235053	0.216928
	4	324	91.36	0	81	8	4.509482	0.434682
	5	389	93.83	0	84	5	4.814591	0.373192
	10	413	83.29	0	51	7	12.213601	1.728035
50	5	490	93.06	0	78	9	3.951199	0.374512
	10	461	87.42	0	65	5	20.524624	1.057270
60	5	569	95.61	0	79	5	7.773736	0.510159
	10	554	91.34	0	66	6	10.940594	0.846808
70	5	715	95.66	0	77	4	3.125759	0.413271
	10	650	94.00	0	73	3	2.815311	0.485601
80	5	806	97.15	0	81	8	9.777588	0.338340
	10	654	91.44	0	61	5	5.004590	0.742792
90	5	821	96.22	0	79	4	4.352552	0.338179
	10	771	92.74	0	64	3	3.976014	0.647212
100	5	984	96.24	0	73	6	2.246622	0.350034
	10	714	92.86	0	63	7	3.588353	0.444596

Tables 2–4 present the percentage of small problem instances which were solved exactly or approximately in off-line phase (by Algorithm 1) and in on-line phase (by Algorithm 2 or 3) in spite of the uncertain numerical input data. Column 1 (column 2) presents number of jobs n, $10 \le n \le 100$ (relative error of the input data L, $1 \le L \le 10$, in percentage).

Column 3 presents the number of sets of conflicting jobs in the strict order $\prec$ for Algorithm 2 with $\mathcal{J}_0 = \emptyset$ (in the binary relation $\preceq$ for Algorigthm 3 with $\mathcal{J}_0 \neq \emptyset$) for which decisions were made in decision-making time-points $t_i = c_1(i)$. (In the above description of Algorithms 2 and 3 this number is denoted as k.) The percentage of the correct decisions made due to Theorem 2 and Propositions 10–15 in the off-line scheduling phase and the correct decisions made due to Propositions 2–11 in the on-line scheduling phase is given in column 4. This number is equal to $\mathrm{m/k} \cdot 100\%$ where m and k are those used in Algorithms 2 and 3.

Table 5
Percentage of solved instances with empty set $\mathcal{J}_0$

n	L (%)	Number of decision points	Percentage of proved decisions	On-line optimal (%)	Optimal without proof (%)	Max error of $C_{\max}$ (%)	Average error of $C_{\max}$ (%)
200	5	1665	96.04	66	28	0.000819	0.000032
	10	909	88.67	42	45	1.687293	0.019191
	15	563	77.09	12	60	1.810943	0.054092
	20	424	64.62	0	66	5.189355	0.219553
300	5	1617	96.23	65	29	0.403977	0.004053
	10	891	85.52	25	59	13.129663	0.131382
	15	548	69.16	1	65	2.639358	0.096203
	20	419	56.09	0	59	5.012075	0.096354
400	5	1605	93.21	47	45	0.000692	0.000014
	10	840	80.48	10	62	3.807984	0.064929
	15	484	65.70	0	68	1.623536	0.025619
	20	361	48.20	0	55	2.821229	0.079312
500	5	1834	95.58	57	31	0.000462	0.000024
	10	840	79.29	4	73	11.960889	0.123434
	15	468	63.03	0	58	5.103044	0.069323
	20	309	33.98	0	44	1.205535	0.025511
600	5	1659	93.31	45	45	2.989783	0.054760
	10	783	77.01	1	60	1.114273	0.026779
	15	417	57.31	0	55	0.212790	0.003442
	20	273	28.21	0	54	0.607854	0.011057
700	5	1766	94.11	48	35	1.544116	0.024961
	10	706	77.48	1	61	1.575914	0.028462
	15	392	51.79	0	55	1.496273	0.027993
	20	244	24.59	0	30	1.170049	0.020081
800	5	1665	92.19	45	39	1.034108	0.011902
	10	691	75.69	0	59	3.810499	0.050601
	15	333	40.24	0	42	2.311629	0.088997
	20	209	20.10	0	36	0.263040	0.002976
900	5	1599	90.43	35	46	6.364723	0.075417
	10	628	67.83	0	54	4.828169	0.068645
	15	323	42.41	0	40	0.438542	0.005272
	20	193	11.92	0	27	2.273839	0.067102
1000	5	1621	92.23	32	49	0.000402	0.000020
	10	593	66.78	1	57	0.000558	0.000062
	15	297	33.33	0	42	3.157275	0.031717
	20	171	14.04	0	27	0.889870	0.023861

The percentage of problem instances which were optimally solved in off-line scheduling phase is given in column 5. For such instances, Algorithm 1 terminates at step 8. The percentage of the problem instances which were optimally solved in on-line scheduling phase (and optimality of the adopted permutation became only known after schedule execution) is given in column 6. For such instances, Algorithm 2 (Algorigthm 3) terminates at step 14. Note that both columns 5 and 6 define the percentage of problem instances for which optimal permutations were defined *before execution* of the whole schedule, i.e., each decision in on-line phase (resolution of the conflicting jobs) was made correctly due to one of the sufficient conditions proven in Sections 4–7.

On the contrary, column 7 presents the percentage of problem instances which were optimally solved occasionally (without a preliminary proof of permutation optimality). Namely, the value $C_{\max}$ obtained for the actual schedule turns out to be equal to the optimal value $C^*_{\max}$ calculated for optimal schedule with the actual job processing times. (Remind that value $C^*_{\max}$ can be calculated after completing the last job from set $\mathcal{J}$ when all actual job processing times $p^*_{ij}, J_i \in \mathcal{J}, M_j \in \mathcal{M}$, and all actual job completion times become known.) For such instances, Algorithm 2 (Algorigthm 3) terminates at step 13. Subtracting the sum of the numbers given in columns 5, 6 and 7 from 100% gives the percentage of instances for which optimal permutations were not found both in off-line scheduling phase and in on-line scheduling phase (for such instances, Algorithms 2 and 3 terminate at step 12). Maximal (average) relative error of the makespan $[(C_{\max} - C^*_{\max})/C^*_{\max}] \cdot 100\%$ obtained for the actual schedule constructed by Algorithms 1 and 2 (Algorigthm 3) is given in column 8 (column 9).

Table 2 presents computational results for the case $\mathcal{J}_0 = \emptyset$ obtained by Algorithms 1 and 2. Table 3 (Table 4) presents computational results for instances with 10% (30%) of the jobs from set $\mathcal{J}_0$ obtained by Algorithms 1 and 3.

Table 5 (Table 6) presents the percentage of large instances ($200 \le n \le 1000$) solved exactly or approximately in off-line phase by Algorithm 1 and in on-line phase by Algorithm 2 for $\mathcal{J}_0 = \emptyset$ (by Algorigthm 3 for $\mathcal{J}_0 \neq \emptyset$). In Tables 5 and 6, we use the same columns as in Tables 2–4 except column 5 since no large instance with $n > 100$ has been optimally solved at off-line phase of scheduling.

Table 6
Percentage of solved instances with 30% of jobs from set $\mathcal{J}_0$

n	L (%)	Number of decision points	Percentage of proved decisions	On-line optimal (%)	Optimal without proof (%)	Max error of $C_{\max}$ (%)	Average error of $C_{\max}$ (%)
	5	1307	95.87	68	7	1.493803	0.228527
200	10	836	92.94	59	10	4.797616	0.268801
	15	592	81.25	23	14	3.531275	0.550724
	20	424	69.58	1	32	8.917567	0.603160
	5	1636	96.45	67	1	6.174240	0.230672
300	10	880	89.20	40	11	2.548966	0.272686
	15	578	77.34	5	27	0.684393	0.327328
	20	448	66.74	0	23	2.098592	0.402790
	5	1612	96.09	60	4	4.655282	0.200383
400	10	873	87.06	28	13	2.560096	0.274370
	15	550	76.18	3	28	3.529257	0.306747
	20	411	61.07	0	21	4.233299	0.330161
	5	1742	95.24	58	2	1.803466	0.155213
500	10	797	84.32	17	27	5.349672	0.271104
	15	526	70.53	0	26	4.218193	0.215071
	20	384	63.28	0	24	3.550418	0.235060
	5	1710	95.50	60	5	1.594194	0.115055
600	10	861	85.25	13	20	1.047556	0.172209
	15	479	68.68	0	32	1.667378	0.192229
	20	335	55.22	0	23	2.664976	0.199377
	5	1787	96.59	62	3	0.283844	0.085372
700	10	821	82.58	8	25	0.974754	0.136061
	15	458	68.12	0	21	3.935263	0.207185
	20	294	46.26	0	30	1.589900	0.154975
	5	1789	95.03	52	8	0.251936	0.081683
800	10	832	83.17	5	24	0.247568	0.125456
	15	443	64.79	0	22	0.766673	0.121888
	20	293	49.49	0	15	1.834536	0.151659
	5	1737	95.28	57	7	0.934157	0.074773
900	10	744	77.42	2	30	0.522401	0.117334
	15	397	59.19	0	28	1.861207	0.141729
	20	263	49.43	0	20	3.329657	0.142571
	5	1724	95.01	53	7	1.111841	0.072543
1000	10	728	80.91	2	28	0.266260	0.098637
	15	404	61.39	0	27	2.376391	0.107110
	20	229	43.23	0	21	1.266256	0.124885

9. Conclusions

We would like to mention that there is another scheduling research line dealing with uncertain processing times, e.g., in [23–25] with a decision criterion of achieving a specified level or even worst-case level and in [26,27] with a decision criterion of minimizing the worst-case regret. Basically, this scheduling research line deals with the off-line phase only. In this research line, one aims to seek one schedule that is optimal from a decision criterion and no attempts are made to take advantage of the local on-line information to best execute the schedule as the scheduled process goes on.

As a new scheme dealing with uncertainty, our scheme must be tested on a representative class of uncertain scheduling problems. To this end in Section 8, two-phase scheduling was tested on a large number of randomly generated problems $F2|p_{ij}^L \le p_{ij} \le p_{ij}^U|C_{\max}$. And the computational results seems to be rather promising, especially for on-line scheduling phase. Tables 2–4 show that the off-line scheduling allowed us to find optimal schedules only for small numbers of jobs and small errors of input data, e.g., for $n = 40$ and $L = 1\%$ dominant permutations have been only obtained for 4% of randomly generated instances. For $n > 40$ there were no such instances at all. Fortunately, on-line scheduling allowed us to find optimal schedules (with optimality proofs before schedule execution) for most instances with $n \le 100$ (Tables 2–4) and for many instances with $200 \le n \le 1000$ (Tables 5 and 6).

The following computational results are even more impressive. The average relative error of the makespan $[(C_{\max} - C^*_{\max})/C^*_{\max}] \cdot 100\%$ obtained for all actual schedules is less than 2.9% for all randomly generated instances with $n = 10$ jobs (column 9 in Tables 2–4). The average relative error of the makespan obtained for all actual schedules is less than 1.67% for all randomly generated instances with n jobs with $20 \le n \le 1000$ (column 9 in Tables 2–4, column 8 in Tables 5 and 6). These results are obtained since the percentage of the correct decisions made in on-line scheduling phase is rather high (column 4). Thus, the sufficient conditions for the existence of a dominant permutation given in Propositions 2–12 may be very effective for on-line scheduling.

It also should be noted that the number of decision-making time-points $t_i = c_1(i)$ when the order of conflicting jobs has to be decided is rather high for some instances with $n \geq 50$ (column 3). However, these decisions made in Algorithm 2 (Algorigthm 3) are very fast: there was no randomly generated instance which takes a running time more than 0.05 s for a processor with 1200 MHz.

Acknowledgments

The research of the first and second authors was supported by INTAS (project 03-51-5501) and ISTC (project B-986).

References

[1] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Prentice-Hall, Englewood Cliffs, USA, 1995.
[2] S. Elmaghraby, K.A. Thoney, Two-machine flowshop problem with arbitrary processing time distributions, IIE Transactions 31 (2000) 467–477.
[3] J. Kamburowski, Stochastically minimizing the makespan in two-machine flow shops without blocking, European Journal of Operational Research 112 (1999) 304–309.
[4] P.S. Ku, S.C. Niu, On Johnson's two-machine flow-shop with random processing times, Operations Research 34 (1986) 130–136.
[5] V. Portougal, D. Trietsch, Johnson's problem with stochastic processing times and optimal service level, European Journal of Operational Research 169 (2006) 751–760.
[6] A. Allahverdi, Stochastically minimizing total flowtime in flowshops with no waiting space, European Journal of Operational Research 113 (1999) 101–112.
[7] A. Allahverdi, J. Mittenthal, Two-machine ordered flowshop scheduling under random breakdowns, Mathematical and Computer Modelling 20 (1994) 9–17.
[8] 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.
[9] 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 (1979) 287–326.
[10] K. Jansen, V. Mastrolilli, R. Solis-Oba, Approximation schemes for job shop scheduling problems with controllable processing times, European Journal of Operational Research 167 (2005) 297–319.
[11] A. Janiak, General flow-shop scheduling with resource constraints, International Journal of Production Research 26 (1988) 125–138.
[12] C.T. Ng, T.C.E. Cheng, M.Y. Kovalyov, Single machine batch scheduling with jointly compressible setup and processing times, European Journal of Operational Research 153 (2004) 211–219.
[13] T.C.E. Cheng, M.Y. Kovalyov, N.V. Shakhlevich, Scheduling with controllable release dates and processing times: Makespan minimization, European Journal of Operational Research 175 (2006) 751–768.
[14] T.-C. Lai, Yu.N. Sotskov, Sequencing with uncertain numerical data for makespan minimization, Journal of the Operational Research Society 50 (1999) 230–243.
[15] N.M. Leshchenko, Yu.N. Sotskov, Two-machine minimum-length shop-scheduling problems with uncertain processing times, in: Proceedings of XI-th International Conference "Knowledge-Dialogue-Solution", vol. 2, Varna, Bulgaria, June 20–0, 2005 pp. 375–381.
[16] T.-C. Lai, Yu.N. Sotskov, N. Sotskova, F. Werner, Mean flow time minimization with given bounds of processing times, European Journal of Operational Research 159 (2004) 558–573.
[17] 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.
[18] 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.
[19] 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.
[20] N.M. Leshchenko, Yu.N. Sotskov, Realization of an optimal schedule for the two-machine flow-shop with interval job processing times, International Journal "Information Theories and Applications" 14 (2) (2007) 182–189.
[21] S.M. Johnson, Optimal two- and three-stage production schedules with setup times included, Naval Research Logistics Quarterly 1 (1954) 61–68.
[22] E. Taillard, Benchmarks for basic scheduling problems, European Journal of Operational Research 64 (1993) 278–285.
[23] R.L. Daniels, P. Kouvelis, Robust scheduling to hedge against processing time uncertainty in single-stage production, Management Science 41 (2) (1995) 363–376.
[24] S.D. Wu, E.-S. Byeon, R. Storer, A graph-theoretical decomposition of the job-shop scheduling problem to achieve scheduling robustness, Operations Research 47 (1) (1999) 113–124.
[25] 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.
[26] I. Averbakh, Minmax regret solutions for minmax optimization problems with uncertainty, Operations Research Letters 27 (2000) 57–65.
[27] 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.