Article:
Brucker, P and Shakhlevich, NV (2011) Inverse scheduling: two machine flow shop problem. Journal of Scheduling, 14 (3). 239 - 256. ISSN: 1094-6136

https://doi.org/10.1007/s10951-010-0168-y

Inverse Scheduling: Two Machine Flow Shop Problem

Peter Brucker
Universität Osnabrück, Fachbereich Mathematik/Informatik,
49069 Osnabrück, Germany (pbrucker@uni-osnabrueck.de)

Natalia V. Shakhlevich*
School of Computing, University of Leeds,
Leeds LS2 9JT, U.K. (N.Shakhlevich@leeds.ac.uk)

Abstract

We study an inverse counterpart of the two machine flow-shop scheduling problem that arises in the context of inverse optimization. While in the forward scheduling problem all parameters are given and the objective is to find job sequence(s) for which the value of the makespan is minimum, in the inverse scheduling the exact values of processing times are unknown and they should be selected within given boundaries so that pre-specified job sequence(s) become optimal. We derive necessary and sufficient conditions of optimality of a given solution for the general case of the flow shop problem when the job sequences on the machines can be different. Based on these conditions we prove that the inverse flow-shop problem is NP-hard even in the case of the same job sequence on both machines and produce a linear programming formulation for a special case which can be solved efficiently.
Keywords: inverse scheduling, flow shop scheduling

1 Introduction

In this paper we study the classical two-machine flow-shop scheduling problem from the *inverse optimization* perspective. While in a forward optimization problem traditionally considered in discrete optimization, the exact values of all parameters of the problem are given and the goal is to find a solution within the solution space with the smallest value of the objective function, in an inverse optimization problem the typical values of the problem parameters are given together with the description of a target, usually non-optimal solution. The objective is to adjust the parameters within certain limits and not deviating too much from their typical values so that the target solution becomes optimal.

Many classical optimization problems have been studied from the point of view of inverse optimization; the summary of the results can be found in the comprehensive reviews [1, 6] and in monograph [13]. The area continues to attract attention of researchers, see, e.g., more recent papers [4, 12, 14].

The adjustable parameters of inverse problems can be of two types. Often the coefficients of the objective function are adjustable (for example, the costs in the assignment

*Correspondence to: Natalia Shakhlevich, e-mail: N.Shakhlevich@leeds.ac.uk

problem). In some research, the adjustable parameters are not related to the objective function as, for example, in the inverse counterpart of the minimum cost flow problem with adjustable capacities, see [5] for the latest study. Inverse problems that arise in the area of scheduling are usually of the second type: the adjustable parameters are various job characteristics such as processing times [10], due dates or release times [3], rather than costs in the objective function.

In this paper we study the inverse counterpart of the two-machine flow shop problem with the makespan objective. In the forward problem denoted by $F2||C_{\max}$, the jobs of the set $N = \{1, 2, \ldots, n\}$ should be processed first by machine A and then by machine B. All jobs are available at time 0. The processing times of the two operations of a job $j \in N$ on machines A and B are given by integers a_j and b_j, respectively. We denote the two vectors of processing times by $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$.

If the jobs are processed in the same order on both machines, then the schedule is called a *permutation schedule.* If additionally the jobs are renumbered so that $\pi = (1, 2, \ldots, n)$, then the makespan $C_{\max}$ of such a schedule can be calculated as

$$C_{\max}(\pi, \pi, \mathbf{a}, \mathbf{b}) = \max_{1 \le h \le n} \left\{ \sum_{j=1}^{h} a_j + \sum_{j=h}^{n} b_j \right\}. \tag{1}$$

In the notation of $C_{\max}$, the first two parameters stay for the permutations on machines A and B while the last two parameters denote the vectors of processing times on those machines.

If the jobs are processed in accordance with different permutations $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ and $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$, then the schedule is called a *non-permutation schedule* and the makespan of such a schedule can be calculated as

$$C_{\max}(\pi, \sigma, \mathbf{a}, \mathbf{b}) = \max_{1 \le h \le n} \left\{ \sum_{j=1}^{\pi^{-1}(h)} a_{\pi(j)} + \sum_{k=\sigma^{-1}(h)}^{n} b_{\sigma(k)} \right\}, \tag{2}$$

where $\pi^{-1}(h)$ and $\sigma^{-1}(h)$ are the positions of job h in permutations π and σ, respectively.

The objective of the forward problem $F2||C_{\max}$ is to find permutations π^* and σ^* for which the makespan is minimum:

$$C_{\max}(\pi^*, \sigma^*, \mathbf{a}, \mathbf{b}) \le C_{\max}(\pi, \sigma, \mathbf{a}, \mathbf{b}) \quad \text{for any job permutations } \pi, \sigma.$$

Observe that allowing different job permutations on the machines cannot decrease the optimal value of the makespan, so that there always exists an optimal permutation schedule. On the other hand, there may also exist an optimal non-permutation schedule with the same value of the makespan.

In the inverse counterpart of problem $F2||C_{\max}$, the typical processing times $\mathbf{a}$ and $\mathbf{b}$ are given together with the target job sequences π and σ on machines A and B, which may be the same on both machines or different. In what follows we always assume that that the jobs are renumbered in accordance with permutation π, so that $\pi = (1, 2, \ldots, n)$. The target job sequences may not be optimal for the given typical values of a_j and b_j, $j \in N$. The objective is to modify the processing times within certain limits so that the target job sequences become optimal.

We denote the inverse problem by $F2|\text{adjustable } a_j, b_j,\ \pi, \sigma|C_{\max}$. In this problem, the adjusted processing times $\widehat{\mathbf{a}} = (\widehat{a}_1, \widehat{a}_2, \ldots, \widehat{a}_n)$ and $\widehat{\mathbf{b}} = \left(\widehat{b}_1, \widehat{b}_2, \ldots, \widehat{b}_n\right)$ should be selected within given boundaries $\widehat{a}_j \in \left[\underline{a}_j,\ \overline{a}_j\right]$, $\widehat{b}_j \in \left[\underline{b}_j,\ \overline{b}_j\right]$, $j \in N$, so that the deviation $\left\|\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right) - (\mathbf{a}, \mathbf{b})\right\|$ from the original processing times is minimum and the target job permutations π and σ for machines A and B are optimal:

$$\begin{array}{lll} \min & \left\|\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right) - (\mathbf{a}, \mathbf{b})\right\| & \\ \text{s.t.} & C_{\max}\left(\pi, \sigma, \widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right) \le C_{\max}\left(\pi', \sigma', \widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right) & \text{for any permutations } \pi', \sigma', \\ & \underline{a}_j \le \widehat{a}_j \le \overline{a}_j, & j \in N, \\ & \underline{b}_j \le \widehat{b}_j \le \overline{b}_j, & j \in N. \end{array} \tag{3}$$

In this paper, the deviation $\left\|\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right) - (\mathbf{a}, \mathbf{b})\right\|$ is estimated in accordance with the norm ℓ_1, which is a popular metric in inverse optimization:

$$\begin{aligned} \left\|\left(\widehat{\mathbf{a}}, \widehat{\mathbf{b}}\right) - (\mathbf{a}, \mathbf{b})\right\|_{1,\alpha,\beta} = & \sum_{j=1}^{n}\left[\alpha_j^+ \max\left\{\widehat{a}_j - a_j, 0\right\} + \alpha_j^- \max\left\{a_j - \widehat{a}_j, 0\right\}\right] \\ & + \sum_{j=1}^{n}\left[\beta_j^+ \max\left\{\widehat{b}_j - b_j, 0\right\} + \beta_j^- \max\left\{b_j - \widehat{b}_j, 0\right\}\right]. \end{aligned}$$

Here coefficients α_j^+, α_j^-, β_j^+ and β_j^- are non-negative. We say that operation of job j on machine A (machine B) is decompressed if $\widehat{a}_j \ge a_j$ ($\widehat{b}_j \ge b_j$) and compressed, otherwise.

Observe that an inverse counterpart of problem $F2||C_{\max}$ is studied in [10] under some additional restrictive conditions. In that study, it is assumed that not only the job sequences are the same on both machines, but a job h that specifies the makespan in (1) is also known. In our study the most general case of the inverse flow-shop problem is considered.

The inverse flow shop problem can be illustrated with the following scenario. Suppose the production process requires some special setups which a producer can perform in advance. If a customer placing several orders can specify only the estimates of job processing times, the producer may plan the production process selecting the best sequences of jobs on the machines based on the information provided and perform the required setups for the selected sequences. The arriving jobs may have slightly different characteristics so that the selected sequences are no longer optimal. If the pre-planned sequences cannot be changed due to technological restrictions, the producer may decide to adjust job processing, speeding up some of them by using, for example, additional resources, or slowing down others, so that the sequences become optimal for the adjusted processing parameters. This, however, incurs costs which should be minimized. Observe that a non-permutation schedule with different job sequences on the machines may be a preferred production plan for the producer; then the required adjustments of processing times are aimed at making the given job sequences optimal.

A similar but slightly different scenario is typical for scheduling with controllable processing times and reverse optimization. In those models, the target value of the objective function is given, while the problem parameters and the solution itself (e.g., the job sequence) should be modified in order to achieve that target value.

The main contributions of the paper can be described as follows. We formulate necessary and sufficient conditions of optimality of a solution given by the same permutation on both

machines and generalize it for the case when the job sequences on the machines can be different. Some constraints of the necessary and sufficient conditions are disjunctive and due to this the inverse flow-shop problem appears to be NP-hard even in the case of the same permutation on both machines. On the other hand, if additional restrictions are imposed, e.g., operations on one machine are fixed and others are adjustable, then the disjunctive constraints can be simplified resulting in a linear programming formulation of the inverse problem.

The paper is organized as follows. The necessary and sufficient conditions are formulated in Section 2. NP-hardness of the inverse flow-shop problem is proved in Section 3. A special case with one adjustable machine is studied in Section 4. Finally, conclusions are given in Section 5.

2 Necessary and Sufficient Conditions of Optimality

In this section we formulate necessary and sufficient conditions of optimality of a solution given by job sequences π and σ on machines A and B. First we consider the permutation schedules in which the job order on both machines is the same ($\pi = \sigma$), then we proceed with the general case when job permutations are different for the two machines ($\pi \neq \sigma$).

2.1 Permutation Schedules

Suppose a target solution is given by a job permutation $\pi = (1, 2, \ldots, n)$ which is the same for machines A and B. We introduce the following notation for cumulative processing times of consecutive operations $u, u+1, \ldots, v$ processed by machines A and B:

$$\begin{aligned} \mathcal{A}_{u,v} &= \sum_{j=u}^{v} a_j, \\ \mathcal{B}_{u,v} &= \sum_{j=u}^{v} b_j, \end{aligned}$$

where $1 \leq u \leq v \leq n$. In what follows, we do not use π, σ, $\mathbf{a}$ and $\mathbf{b}$ in the notation of $C_{\max}$ if no ambiguity arises.

Definition 1 *Job $h \in N$ is* critical *if*

$$C_{\max} = \mathcal{A}_{1,h} + \mathcal{B}_{h,n} \tag{4}$$

or equivalently

$$\mathcal{A}_{1,h} + \mathcal{B}_{h,n} \geq \mathcal{A}_{1,j} + \mathcal{B}_{j,n} \tag{5}$$

for all $j \in N$.

Observe that the notion of a critical job plays an important role in the flow shop problem, see, e.g., [11]. Its meaning can be explained by using the network representation $G = (V, E)$ of a flow-shop schedule given by the job sequence $\pi = (1, 2, \ldots, n)$ on machines A and B. In that network, the vertices represent the operations of the jobs N on machines A and B plus the source s and the terminal node t, $|V| = 2n + 2$. The arcs E represent the precedence relations among the operations:

- in accordance with permutation $\pi = (1, 2, \ldots, n)$, every pair of nodes j and j+1 associated with machine A are connected by an arc; similarly, every pair of nodes j and $j + 1$ associated with machine B are also connected by an arc;

- in accordance with the flow-shop requirement, for every job j, its operation on machine A should precede its operation on machine B;

- source s is connected with the first operation of job 1 on machine A, while the last operation of job n on machine B is connected to the terminal node t.

The length of the arc $(s, 1)$ is zero, while for any other arc its length is defined as the processing time of an operation the arc originates from.

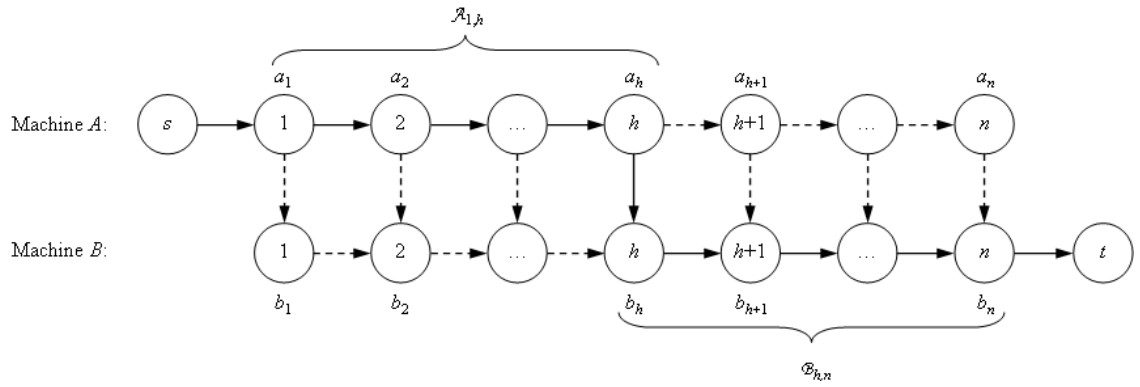

Figure 1: Network model $G = (V, E)$ for the flow-shop schedule with the same permutation $\pi = (1, 2, \ldots, n)$ on both machines

Using the network representation G, the starting time of any operation can be found as the length of the longest path from s to that operation, and the makespan $C_{\max}$ corresponds to the starting time of t or equivalently to the length of the longest (critical) path from s to t in G. Clearly, it should contain one arc connecting two operation-nodes of the same job, which we call critical and denote by h, and the length of the critical path is calculated in accordance with (4). In the network model, $\mathcal{A}_{1,h}$ is the length of the path on machine A from s to the A-operation of job h plus the length of the arc a_h connecting the two operations of job h, and $\mathcal{B}_{h,n}$ is the length of the path on machine B from the B-operation of job h to t.

An example of the network representation is shown in Fig. 1. It is assumed that the longest path, shown by solid arcs, passes through the two nodes of job h, which is a critical job.

The role of machines is interchangeable in the flow shop problem, and for any schedule with machine order A, B and job order $\pi = (1, 2, \ldots, n)$ there exists a *symmetric counterpart* with machine order B, A and reverse job order $(n, n-1, \ldots, 1)$. It is easy to see that the corresponding network model can be easily modified for that counterpart by reversing all arcs. The two schedules are equivalent in the sense that the critical path passes through the same critical job h in both networks and therefore the makespan value is the same.

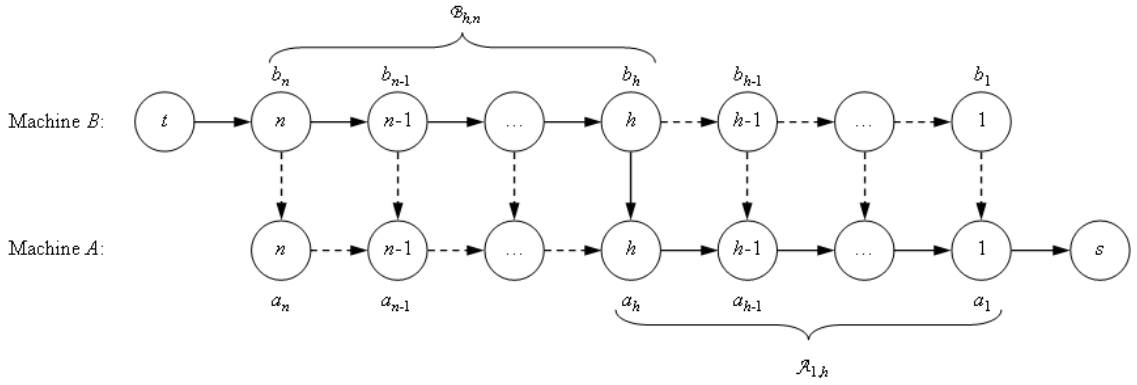

Figure 2: Network model $G = (V, E)$ for the symmetric counterpart with the reverse order of jobs and reverse order of machines

Consider separately jobs $u \in \{1, 2, \ldots, h-1\}$, which precede the critical job h, and jobs $v \in \{h+1, \ldots, n\}$, which follow it. Then conditions (5) are equivalent to

$$\mathcal{A}_{u,h} \geq \mathcal{B}_{u-1,h-1}, \quad 2 \leq u \leq h, \tag{6}$$

$$\mathcal{A}_{h+1,v} \leq \mathcal{B}_{h,v-1}, \quad h+1 \leq v \leq n. \tag{7}$$

Notice that we have no constraint (6) if $h = 1$ and symmetrically there is no constraint (7) if $h = n$.

In what follows we formulate necessary and sufficient conditions of optimality of a given permutation π.

Theorem 1 *[11] The job sequence $\pi = (1, 2, \ldots, n)$ is optimal if and only if there is a critical job $h \in N$ such that conditions (6)-(7) hold and*

$$\min\{a_u, b_v\} \leq \min\{b_u, a_v\} \quad \text{for all } u \leq h \leq v. \tag{8}$$

Observe that there may be several critical jobs in an optimal schedule and it may happen that only for some of them the conditions of Theorem 1 hold while for the other critical jobs they are not satisfied. An example of such a schedule will be provided in Section 4.1 with two critical jobs, one of which satisfies the conditions of Theorem 1 and another one does not.

We reformulate conditions (8) as inequalities which can be used in mathematical programming problem (3). In addition, we also include the relevant conditions (6)-(7) which guarantee that job h is critical. Since we do not know in advance which job from the set N should become critical for the adjusted processing times, we enumerate different classes of schedules with the fixed critical job h and find optimum adjustments in each class; the global solution is selected among the optimum solutions found in each class ensuring that the adjustment cost is minimum.

Let the set of jobs N be split into three subsets:

$$\begin{aligned} N_0 &= \{j | a_j = b_j\}, \\ N_1 &= \{j | a_j < b_j\}, \\ N_2 &= \{j | a_j > b_j\}. \end{aligned}$$

In the following two theorems we consider the two cases: $h \in N_1$ and $h \in N_0$; the case $h \in N_2$ can be reduced to the case $h \in N_1$ by formulating a symmetric counterpart of the flow shop problem.

We start with the case $h \in N_1$.

Theorem 2 *Suppose there exists an optimal schedule with a critical job $h \in N_1$. Then a schedule given by permutation $\pi = (1, 2, \ldots, n)$ with that critical job is optimal if and only if conditions (6)-(7) hold and*

- *for each $u \in \{1, \ldots, h-1\}$,*

$$a_u \leq b_u \quad \text{and} \quad a_u \leq a_h \tag{9}$$

- *for each $v \in \{h+1, \ldots, n\}$, either*

$$a_h \leq a_v < b_v \tag{10}$$

or

$$a_v \geq b_v. \tag{11}$$

Proof. First we verify that the conditions of Theorem 2 imply (8). Clearly, (8) is satisfied if $u = v$. If $u < v$, then we have the following three cases.

(i) Condition (9) together with $a_h < b_h$ imply that (8) is satisfied for $u < h = v$ because $a_u \leq a_h = a_v$ and $a_u \leq b_u$.

(ii) Condition $a_h < b_h$ together with one of the conditions (10) or (11) imply that (8) is satisfied for $u = h < v$. Indeed, in case of (10), $a_u = a_h \leq a_v$ and $a_u = a_h < b_h = b_u$, so that (8) holds. In case of (11), $b_v \leq a_v$ and $a_u = a_h < b_h = b_u$, so that (8) holds as well.

(iii) Condition (9) together with one of the conditions (10) or (11) imply that (8) is satisfied for $u < h < v$. Indeed, (9) and (10) imply $a_u \leq b_u$ and $a_u \leq a_h \leq a_v$, so that (8) holds. Similarly, (9) and (11) imply $a_u \leq b_u$ and $b_v \leq a_v$, so that (8) holds as well.

In what follows we demonstrate that conditions (8) imply the conditions of Theorem 2. In particular, we show that condition (8) with $u < h = v$ imply (9) (see Part A) and condition (8) with $u = h < v$ imply (10)-(11) (see Part B).

Part A. Suppose

$$a_h < b_h, \tag{12}$$

$$\min\{a_u, b_h\} \leq \min\{a_h, b_u\}. \tag{13}$$

First we observe that no job u with $a_u > b_u$ can satisfy (13). Indeed, if this was a case, then combining $a_u > b_u$ with (12) we obtain $\min\{a_u, b_h\} > \min\{a_h, b_u\}$, a contradiction to (13).

In addition, the inequality $b_h < a_u$ can never happen: if this was a case, then $a_h < b_h < a_u \leq b_u$, so that by (13) we have $b_h = \min\{a_u, b_h\} \leq \min\{a_h, b_u\} = a_h$, a contradiction to (12).

Thus the only possible case is $b_h \geq a_u$. Using this in (13) we obtain: $a_u = \min\{a_u, b_h\} \leq \min\{a_h, b_u\}$, or equivalently $a_u \leq a_h$ and $a_u \leq b_u$, which are the two inequalities from (9).
Part B. Suppose

$$a_h < b_h, \tag{14}$$

$$\min\{a_h, b_v\} \leq \min\{a_v, b_h\}. \tag{15}$$

We demonstrate that either condition (10) or condition (11) holds. If (11) holds, we are done. Otherwise $a_v < b_v$, which together with (15) implies $a_v \geq a_h$. Therefore (10) holds. ■

Observe that if the symmetric counterpart with $h \in N_2$ is considered, then the equivalent formulation of Theorem 2 should have a condition similar to (9) formulated for all jobs $\ell \in \{h+1, \ldots, n\}$ and in that condition a-values are replaced by b-values and vice versa:

$$b_\ell \leq a_\ell \text{ and } b_\ell \leq b_h. \tag{16}$$

Conditions similar to (10)-(11) should be formulated for all jobs $k \in \{1, 2, \ldots, h-1\}$ and in those conditions a-values are also replaced by b-values and vice versa:

$$b_h \leq b_k < a_k \tag{17}$$

or

$$b_k \geq a_k. \tag{18}$$

Now we study the case of a critical job $h \in N_0$. First we observe that if there is at least one job $u \in N_2$ which precedes h, then it is not possible that some job v which follows h belongs to N_1 since for jobs u and v with $a_u > b_u$ and $a_v < b_v$ condition (8) does not hold. Therefore all possible situations are covered by the two cases:

(i) $u \in N_1 \cup N_0$ for all $u < h$;

(ii) $v \in N_2 \cup N_0$ for all $v > h$.

We can consider only situation (i); situation (ii) can be reduced to (i) by considering the symmetric counterpart of the flow shop problem. It appears that in case (i) with $h \in N_0$ the conditions are the same as those formulated in Theorem 2 for $h \in N_1$.

Theorem 3 *Suppose there exists an optimal schedule with a critical job $h \in N_0$ and all jobs before h belonging to $N_1 \cup N_0$. Then a schedule given by permutation $\pi = (1, 2, \ldots, n)$ with a critical job $h \in N_0$ is optimal if and only if condition (9) holds for each $u \in \{1, \ldots, h-1\}$ and one of the conditions (10) or (11) holds for each $v \in \{h+1, \ldots, n\}$.*

Proof. It is easy to verify that the arguments used in the proof of Theorem 2 which show that conditions (8) follow from the conditions of Theorem 2 are applicable for the case $h \in N_0$.

In what follows we demonstrate that conditions (8) imply the conditions of Theorem 3. Similar to part A from the proof of Theorem 2, we show that condition (8) with $u < h = v$ imply (9). Indeed, the inequality $a_u > a_h$ cannot hold; if this was a case, then condition (6) would be violated and job h could not be critical:

$$\mathcal{A}_{u+1,h} < \mathcal{A}_{u+1,h-1} + a_u \leq \mathcal{B}_{u+1,h-1} + b_u = \mathcal{B}_{u,h-1}.$$

Here the second inequality follows from the assumption of the theorem that all jobs preceding h belong to $N_1 \cup N_0$.

The proof that condition (8) with $u = h < v$ imply (10)-(11) repeats the arguments from the proof of Theorem 2. ■

It is well known that an optimal solution to the two-machine flow shop problem can be found in accordance with the algorithm formulated by Johnson [8]. It constructs an optimal flow shop schedule by sequencing the jobs from $N_1 \cup N_0$ in non-decreasing order of their a-values and then the jobs from N_2 in non-increasing order of their b-values. Thus the Johnson rule implies the conditions of Theorems 2-3 but not vice versa, i.e., Johnson's conditions are sufficient for optimality but not necessary. In fact there may exist many optimal schedules which are very different from Johnson's schedule as illustrated in the example below. Not only the jobs in N_0 can be moved to different positions, jobs in N_1 and N_2 can also be moved violating the Johnson sequence, but in accordance with the necessary and sufficient conditions of optimality. Moreover, there may exist optimal schedules with job(s) from N_2 preceding job(s) from N_1.

j	1	2	3	4	5	6	7	8	9
a_j	1	1	2	2	6	7	4	5	3
b_j	3	1	3	2	9	10	3	2	1

Table 1: Input data of an instance of the flow shop problem

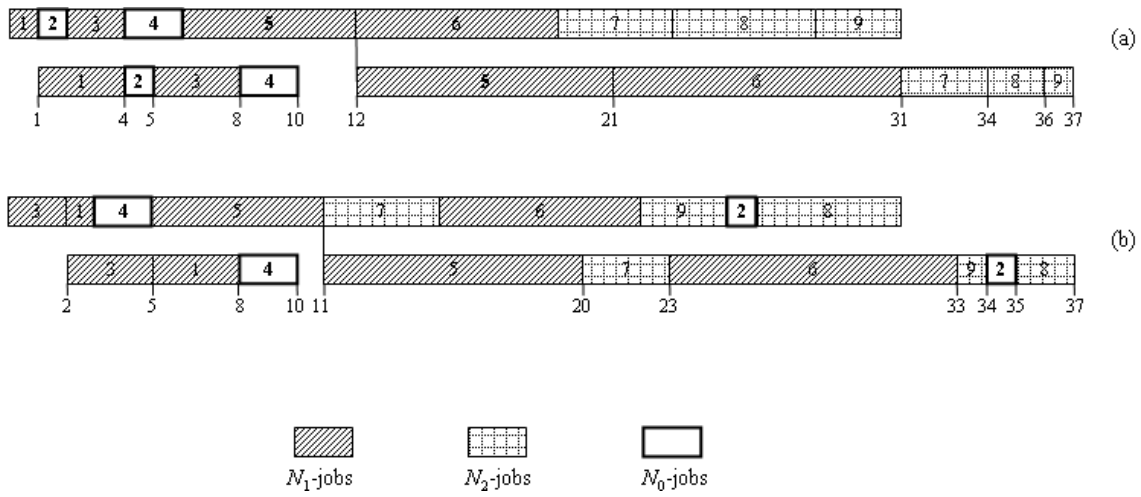

Figure 3: Two optimal schedules satisfying the necessary and sufficient conditions of Theorem 2

Consider an instance of the flow-shop problem with the data given by Table 1 and two optimal schedules shown in Fig. 3 (a) and (b). In the figures, the jobs from N_1 and N_2 are marked by different patterns while the jobs from N_0 are left blank. Both schedules have the same critical job $h = 5$ and the same makespan $C_{\max} = 37$. The schedule shown in

Fig. 3 (a) follows the Johnson rule with the jobs from $N_1 \cup N_0$ preceding the jobs from N_2. The second schedule shown in Fig. 3 (b) is not consistent with the Johnson rule: the positions of N_0-jobs are changed so that N_0-job 2 appears in-between two N_2-jobs; the jobs from N_1 are not sequenced in non-decreasing order of their a-values; the jobs from N_2 are not sequenced in non-increasing order of their b-values; moreover, a job from N_2 (job 7) precedes a job from N_1 (job 6). Still the second schedule satisfies the necessary and sufficient conditions and it is optimal.

2.2 Non-permutation Schedules

We start with the definition of a critical job h in a non-permutation schedule. Then we show that in an optimal schedule job h splits the schedule into two parts with one set of jobs processed on machines A and B before h and the remaining jobs processed after h on both machines, so that the necessary and sufficient conditions formulated for the permutation case are applicable to the non-permutation case.

Suppose a schedule is given by job sequences $\pi = (1, 2, \ldots, n)$ and $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ on machines A and B, respectively. We refine definition (2) of a critical job h for a non-permutation schedule. Let $N_A^{\text{before}(h)}$ and $N_A^{\text{after}(h)}$ ($N_B^{\text{before}(h)}$ and $N_B^{\text{after}(h)}$) denote the jobs processed on machine A (machine B) before and after job h. For operations on machine A, we denote the total sum of all operations in $N_A^{\text{before}(h)}$ and $N_A^{\text{after}(h)}$ by $\mathcal{A}\left(N_A^{\text{before}(h)}\right)$ and $\mathcal{A}\left(N_A^{\text{after}(h)}\right)$; similarly for operations on machine B, we denote the total sum of all operations in $N_B^{\text{before}(h)}$ and $N_B^{\text{after}(h)}$ by $\mathcal{B}\left(N_B^{\text{before}(h)}\right)$ and $\mathcal{B}\left(N_B^{\text{after}(h)}\right)$.

Definition 2 *Job h is called* critical *in a non-permutation schedule if*

$$C_{\max} = \mathcal{A}\left(N_A^{\text{before}(h)}\right) + a_h + b_h + \mathcal{B}\left(N_B^{\text{after}(h)}\right) \tag{19}$$

or equivalently

$$\mathcal{A}\left(N_A^{\text{before}(h)}\right) + a_h + b_h + \mathcal{B}\left(N_B^{\text{after}(h)}\right) \geq \mathcal{A}\left(N_A^{\text{before}(j)}\right) + a_j + b_j + \mathcal{B}\left(N_B^{\text{after}(j)}\right) \tag{20}$$

for any $j \in N$.

The network representation $G = (V, E)$ introduced in Section 2.1, is applicable to the non-permutation case. It illustrates that any path from s to t should contain exactly one arc connecting the two operation-nodes of the same job h, and the length of the critical path is calculated in accordance with (19), where $\mathcal{A}\left(N_A^{\text{before}(h)}\right)$ is the length of the path on machine A from s to the A-operation of job h, $a_h + b_h$ is the total length of the two operations of job h, and $\mathcal{B}\left(N_B^{\text{after}(h)}\right)$ is the length of the path after job h on machine B terminating in t.

An example of the network representation of the schedule given by $\pi = (1, 2, 3, 4, 5, 6, 7)$ and $\sigma = (1, 3, 4, 2, 5, 7, 6)$ is shown in Fig. 4 for the processing times given by Table 2.

The longest path passes through the two nodes of job 4, which is critical, and the makespan is given by $(a_1 + a_2 + a_3) + a_4 + b_4 + (b_2 + b_5 + b_7 + b_6) = 32$.

Now we show that the necessary and sufficient conditions formulated in Theorems 2-3 hold for the non-permutation case.

j	1	2	3	4	5	6	7
a_j	1	4	3	6	2	8	1
b_j	8	2	3	5	6	3	2

Table 2: Input data for an non-permutation schedule

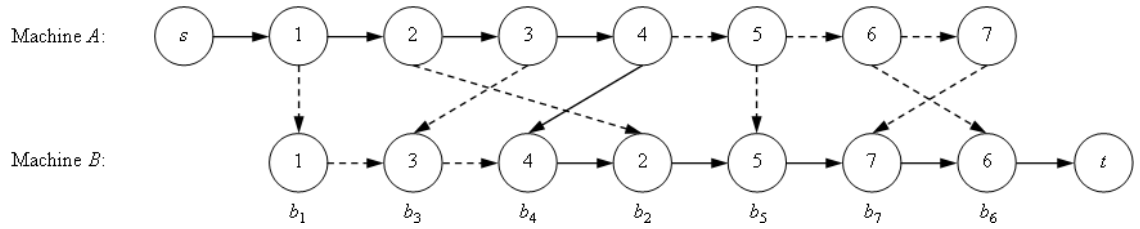

Figure 4: Network representation of the schedule given by $\pi = (1, 2, 3, 4, 5, 6, 7)$ and $\sigma = (1, 3, 4, 2, 5, 7, 6)$

Theorem 4 *A schedule S^* given by permutations $\pi = (1, 2, \ldots, n)$ and $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ on machines A and B is optimal if and only if there exists a critical job h such that it partitions the jobs N into the same subsets on machines A and B:*

$$\{1, 2, \ldots, h-1\} = N_A^{\text{before}(h)} = N_B^{\text{before}(h)} = \{\sigma(1), \sigma(2), \ldots, \sigma(h-1)\}, \quad (21)$$

$$h = \sigma(h), \quad (22)$$

$$\{h+1, \ldots, n\} = N_A^{\text{after}(h)} = N_B^{\text{after}(h)} = \{\sigma(h+1), \ldots, \sigma(n)\}, \quad (23)$$

and the necessary and sufficient conditions, formulated for the permutation case for all $u \in N_A^{\text{before}(h)}$ and $v \in N_A^{\text{after}(h)}$, are satisfied as well.

Observe that it is enough to prove one of the conditions (21) or (23); together with (22), the other condition follows immediately.

Proof. Suppose S^* is an optimal non-permutation schedule. We prove that a critical job exists such that conditions (22) and (23) hold together with the necessary and sufficient conditions, formulated for the permutation case.

Introduce a permutation schedule $\widetilde{S}$ keeping the job order π on machine A the same as in S^* and changing the job order on machine B from σ to π. It is known that $C_{\max}\left(\widetilde{S}\right) \leq C_{\max}(S^*)$, see, e.g., [2]. Due to the optimality of S^*, the above inequality should hold as an equality, and schedule $\widetilde{S}$ is optimal as well.

At least one of the critical jobs in $\widetilde{S}$ satisfies the necessary and sufficient conditions known for the permutation case. Suppose h is such a job in $\widetilde{S}$. Denote by $\widetilde{N}_A^{\text{before}(h)}$ the subset of jobs in $\widetilde{S}$ before job h on machine A and by $\widetilde{N}_B^{\text{after}(h)}$ the subset of jobs in $\widetilde{S}$ after job h on machine B. Clearly,

$$\widetilde{N}_A^{\text{before}(h)} = N_A^{\text{before}(h)}$$

since permutation π is the same on machine A in both schedules $\widetilde{S}$ and S^*. If in addition

$$\widetilde{N}_B^{\text{after}(h)} = N_B^{\text{after}(h)}, \quad (24)$$

then

$$C_{\max}\left(\widetilde{S}\right) = \widetilde{N}_A^{\text{before}(h)} + a_h + b_h + \widetilde{N}_B^{\text{after}(h)} = N_A^{\text{before}(h)} + a_h + b_h + N_B^{\text{after}(h)} \leq C_{\max}\left(S^*\right),$$

where the last inequality holds due to the definition of the makespan in schedule S^*. If that inequality is strict, then $C_{\max}\left(\widetilde{S}\right) < C_{\max}\left(S^*\right)$ and S^* is not optimal, a contradiction. Otherwise, job h is critical in S^* and the correctness of the remaining statements of the theorem follows from the fact that the necessary and sufficient conditions of optimality of schedule $\widetilde{S}$, formulated in Theorems 2-3 involve inequalities for the jobs $\widetilde{N}_A^{\text{before}(h)}$ and $\widetilde{N}_B^{\text{after}(h)}$, the order of the jobs in each of these subsets being immaterial.

Suppose now that condition (24) is not satisfied. We show that in the schedule S^* every job which belongs to $N_A^{\text{after}(h)}$ should also belong to $N_B^{\text{after}(h)}$, so that

$$N_A^{\text{after}(h)} \subset N_B^{\text{after}(h)}. \tag{25}$$

If this is not the case, then consider a job j which satisfies:

$$\begin{aligned} j &\in N_A^{\text{after}(h)} = \widetilde{N}_B^{\text{after}(h)}, \\ j &\notin N_B^{\text{after}(h)}. \end{aligned}$$

If there is more than one job with this property, then among those jobs select job j as the earliest one on machine B. By the choice of j,

$$b_j + \mathcal{B}\left(N_B^{\text{after}(j)}\right) \geq b_h + \mathcal{B}\left(\widetilde{N}_B^{\text{after}(h)}\right).$$

Taking into account that

$$\mathcal{A}\left(N_A^{\text{before}(j)}\right) \geq \mathcal{A}\left(\widetilde{N}_A^{\text{before}(h)}\right) + a_h,$$

we conclude that the path that passes through both operations of job j in the network representation of schedule S^* is longer than the one that passes through both operations of job h in the network representation of schedule $\widetilde{S}$:

$$\begin{aligned} \mathcal{A}\left(N_A^{\text{before}(j)}\right) + a_j + b_j + \mathcal{B}\left(N_B^{\text{after}(j)}\right) &\geq \left[\mathcal{A}\left(\widetilde{N}_A^{\text{before}(h)}\right) + a_h\right] + a_j + \\ &\quad + \left[b_h + \mathcal{B}\left(\widetilde{N}_B^{\text{after}(h)}\right)\right] \\ &> \mathcal{A}\left(\widetilde{N}_A^{\text{before}(h)}\right) + a_h + b_h + \mathcal{B}\left(\widetilde{N}_B^{\text{after}(h)}\right) \\ &= C_{\max}\left(\widetilde{S}\right), \end{aligned}$$

a contradiction to the optimality of the schedule S^*.

In what follows we assume that $N_A^{\text{after}(h)} = \widetilde{N}_B^{\text{after}(h)} \subset N_B^{\text{after}(h)}$. Then

$$\begin{aligned} C_{\max}\left(\widetilde{S}\right) &= \mathcal{A}\left(\widetilde{N}_A^{\text{before}(h)}\right) + a_h + b_h + \mathcal{B}\left(\widetilde{N}_B^{\text{after}(h)}\right) < \\ &< \mathcal{A}\left(N_A^{\text{before}(h)}\right) + a_h + b_h + \mathcal{B}\left(N_B^{\text{after}(h)}\right) \leq C_{\max}\left(S^*\right), \end{aligned}$$

which implies $C_{\max}\left(\widetilde{S}\right) < C_{\max}\left(S^*\right)$, a contradiction to the optimality of schedule S^*. Thus condition (24) holds and we arrive at the case already considered.

Now suppose that in schedule S^* given by $\pi = (1, 2, \ldots, n)$ and $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ there exists a critical job h which satisfies inequalities (21)-(23) and the necessary and sufficient conditions, formulated for the permutation case. Without loss of generality we assume that $h \in N_1$ so that Theorem 2 holds or $h \in N_0$ and all jobs preceding h belong to $N_1 \cup N_0$ so that Theorem 3 holds. We prove that S^* is optimal.

Since in schedule S^* job h is critical and (21)-(23) hold, then

$$\mathcal{A}_{1,h} + \mathcal{B}_{h,n} = \mathcal{A}_{1,h} + b_h + \mathcal{B}\left(N_B^{after(h)}\right) \geq \mathcal{A}_{1,j} + \sum_{k=\sigma^{-1}(j)}^{n} b_{\sigma(k)} \tag{26}$$

for any job $j \in N$, where $\sigma^{-1}(j)$ is the position of job j in permutation σ.

Consider again a permutation schedule $\widetilde{S}$ obtained from S^* keeping the job order π on machine A the same as in S^* and changing the job order on machine B from σ to π.

With (21)-(23) we have

$$\begin{array}{l} \mathcal{A}\left(\widetilde{N}_A^{before(h)}\right) + a_h + b_h + \mathcal{B}\left(\widetilde{N}_B^{after(h)}\right) = \\ \mathcal{A}\left(N_A^{before(h)}\right) + a_h + b_h + \mathcal{B}\left(N_B^{after(h)}\right) = C_{\max}\left(S^*\right). \end{array} \tag{27}$$

If h is critical for $\widetilde{S}$, then the left-hand side of (27) provides the optimal makespan for the corresponding permutation flow shop problem, because all the optimality criteria are satisfied. This value is also the optimal makespan for the non-permutation flow shop problem. Thus S^* is optimal.

In what follows we show that h is always critical for $\widetilde{S}$ by demonstrating that (5) holds for all $j \in N$. To this end, we consider $j \in N$ under each of the following two conditions.

(1) If there is no job which is sequenced after j in π and before j in σ, then

$$\{\sigma(\nu), \sigma(\nu+1), \ldots, \sigma(n)\} \supseteq \{j, j+1, \ldots, n\}$$

where $\nu = \sigma^{-1}(j)$ is the position of job j in σ. Hence

$$\sum_{k=\sigma^{-1}(j)}^{n} b_{\sigma(k)} \geq \sum_{k=j}^{n} b_k.$$

The latter inequality combined with (26) implies condition (5).

(2) If there is a job which is sequenced after j in π and before j in σ, then among the jobs with this property select a job ℓ which is the earliest one in σ. Due to the numbering of jobs in π and due to the fact that ℓ is sequenced after j in π,

$$\{1, 2, \ldots, j\} \subset \{1, 2, \ldots, j, \ldots, \ell\}.$$

Due to the choice of ℓ, any job processed after j in π is processed after ℓ in σ:

$$\{j, j+1, \ldots, n\} \subseteq \{\sigma(\mu), \ldots, \sigma(\nu), \ldots, \sigma(n)\}$$

where $\mu = \sigma^{-1}(\ell)$ and $\nu = \sigma^{-1}(j)$ are the positions of jobs ℓ and j in σ, respectively. Therefore

$$\begin{aligned} \mathcal{A}_{1,j} &< \mathcal{A}_{1,\ell}, \\ \sum_{k=j}^{n} b_k &\leq \sum_{k=\sigma^{-1}(\ell)}^{n} b_{\sigma(k)}. \end{aligned}$$

Since h is critical in schedule S^*, condition (26) holds for $j = \ell$:

$$\mathcal{A}_{1,h} + \mathcal{B}_{h,n} \geq \mathcal{A}_{1,\ell} + \sum_{k=\sigma^{-1}(\ell)}^{n} b_{\sigma(k)}.$$

Combining the last three inequalities we obtain that (5) holds as a strict inequality. ∎

Due to Theorem 4, we can enumerate only those classes of schedules, for which a fixed critical job $h \in N$ splits the jobs on machines A and B in accordance with (21)-(23). In the mathematical programming formulation (3) for a fixed critical job h, the relevant conditions (20) which guarantee that h is critical should be added to the set of constraints. For example, in the instance shown in Fig. 4, there are two jobs 1 and 5 which satisfy (21)-(23). In the class of the schedules with the critical job $h = 5$, the constraints (20) are of the form:

$$\begin{aligned}
(\widehat{a}_1 + \widehat{a}_2 + \widehat{a}_3 + \widehat{a}_4) + \widehat{a}_5 + \widehat{b}_5 + \left(\widehat{b}_7 + \widehat{b}_6\right) &\geq \widehat{a}_1 + \widehat{b}_1 + \left(\widehat{b}_3 + \widehat{b}_4 + \widehat{b}_2 + \widehat{b}_5 + \widehat{b}_7 + \widehat{b}_6\right) \\
(\widehat{a}_1 + \widehat{a}_2 + \widehat{a}_3 + \widehat{a}_4) + \widehat{a}_5 + \widehat{b}_5 + \left(\widehat{b}_7 + \widehat{b}_6\right) &\geq \widehat{a}_1 + \widehat{a}_2 + \widehat{b}_2 + \left(\widehat{b}_5 + \widehat{b}_7 + \widehat{b}_6\right) \\
(\widehat{a}_1 + \widehat{a}_2 + \widehat{a}_3 + \widehat{a}_4) + \widehat{a}_5 + \widehat{b}_5 + \left(\widehat{b}_7 + \widehat{b}_6\right) &\geq (\widehat{a}_1 + \widehat{a}_2) + \widehat{a}_3 + \widehat{b}_3 + \left(\widehat{b}_4 + \widehat{b}_2 + \widehat{b}_5 + \widehat{b}_7 + \widehat{b}_6\right) \\
(\widehat{a}_1 + \widehat{a}_2 + \widehat{a}_3 + \widehat{a}_4) + \widehat{a}_5 + \widehat{b}_5 + \left(\widehat{b}_7 + \widehat{b}_6\right) &\geq (\widehat{a}_1 + \widehat{a}_2 + \widehat{a}_3) + \widehat{a}_4 + \widehat{b}_4 + \left(\widehat{b}_2 + \widehat{b}_5 + \widehat{b}_7 + \widehat{b}_6\right) \\
(\widehat{a}_1 + \widehat{a}_2 + \widehat{a}_3 + \widehat{a}_4) + \widehat{a}_5 + \widehat{b}_5 + \left(\widehat{b}_7 + \widehat{b}_6\right) &\geq (\widehat{a}_1 + \widehat{a}_2 + \widehat{a}_3 + \widehat{a}_4 + \widehat{a}_5) + \widehat{a}_6 + \widehat{b}_6 \\
(\widehat{a}_1 + \widehat{a}_2 + \widehat{a}_3 + \widehat{a}_4) + \widehat{a}_5 + \widehat{b}_5 + \left(\widehat{b}_7 + \widehat{b}_6\right) &\geq (\widehat{a}_1 + \widehat{a}_2 + \widehat{a}_3 + \widehat{a}_4 + \widehat{a}_5 + \widehat{a}_6) + \widehat{a}_7 + \widehat{b}_7 + \widehat{b}_6
\end{aligned}$$

The remaining constraints are of the form (9), (10) or (11) together with the box constraints which specify job variability intervals.

3 NP-hardness of the Inverse Flow Shop Problem

Although we have demonstrated that the same necessary and sufficient conditions of optimality hold for permutation and non-permutation schedules, in what follows we deal with permutation schedules only assuming that the jobs are sequenced in the same order $\pi = (1, 2, \ldots, n)$ on both machines. We show that the corresponding inverse problem is NP-hard by using a reduction from the knapsack problem. The main idea of the reduction is based on the fact that the jobs which follow a critical job should satisfy one of the disjunctive constraints (10) or (11). Therefore for a job that violates both constraints, one of the two possible adjustments can be applied. Selecting one of the two adjustments for such a job can be interpreted in terms of the knapsack problem as selecting an item for including it in the knapsack or rejecting it.

Theorem 5 *Problem* $F2|\text{adjustable } a_j, b_j,\ \pi, \pi|C_{\max}$ *is NP-hard.*

Proof. First notice that inverse problem $F2|\text{adjustable } a_j, b_j,\ \pi, \pi|C_{\max}$ is in NP because with Theorem 2 the optimality of a given solution and whether the adjustment cost is below a given threshold value can be checked in polynomial time.

In order to prove NP-hardness of problem $F2|\text{adjustable } a_j, b_j, \pi, \pi|C_{\max}$ we define an instance of this problem which is equivalent to the knapsack problem

$$\begin{array}{ll} \max & \sum_{i=1}^{q} \gamma_i z_i \\ \text{s.t.} & \sum_{i=1}^{q} w_i z_i \leq C, \\ & z_i \in \{0,1\}, \qquad 1 \leq i \leq q. \end{array} \tag{28}$$

with integers $w_i, \gamma_i > 0$ for all $1 \leq i \leq q$, integer $C > 0$ and

$$\sum_{i=1}^{q} w_i > C.$$

Consider $q+3$ jobs $h, v_1, \ldots, v_q, k, \ell$ given in this order. Three jobs h, k and ℓ are "special" jobs and they cannot be adjusted:

$$\begin{array}{lll} \underline{a}_h = a_h = \overline{a}_h, & \underline{b}_h = b_h = \overline{b}_h, \\ \underline{a}_k = a_k = \overline{a}_k, & \underline{b}_k = b_k = \overline{b}_k, \\ \underline{a}_\ell = a_\ell = \overline{a}_\ell, & \underline{b}_\ell = b_\ell = \overline{b}_\ell. \end{array} \tag{29}$$

Job k is a "big" job from N_2 with

$$\begin{aligned} a_k &= E, \\ b_k &= E - 1, \end{aligned}$$

where E is defined as the total weight of all items for the knapsack plus q:

$$E = \sum_{i=1}^{q} w_i + q. \tag{30}$$

Jobs h and ℓ are from N_1 and they are identical:

$$\begin{aligned} a_h &= a_\ell = 1 + \varepsilon, \\ b_h &= b_\ell = C + \frac{1}{2} + q < E, \end{aligned}$$

where C is the knapsack capacity and ε is a small positive number that satisfies

$$\varepsilon < \frac{1}{2q}. \tag{31}$$

The processing times of the remaining jobs $\{v_1, \ldots, v_q\}$ are defined as

$$\begin{array}{ll} a_{v_i} = \varepsilon, & i = 1, \ldots, q, \\ b_{v_i} = w_{v_i} + 1, & i = 1, \ldots, q. \end{array} \tag{32}$$

The adjustment boundaries for the jobs $\{v_1, \ldots, v_q\}$ are as follows:

$$\underline{a}_{v_i} = a_{v_i}, \quad \overline{a}_{v_i} = a_h, \quad i = 1, \ldots, q, \quad (a_{v_i} \text{ cannot be compressed but can be decompressed up to } a_h)$$
$$\underline{b}_{v_i} = a_{v_i}, \quad \overline{b}_{v_i} = b_{v_i}, \quad i = 1, \ldots, q, \quad (b_{v_i} \text{ cannot be decompressed but can be compressed down to } a_{v_i})$$

and the adjustment costs satisfy

$$\alpha^+_{v_i} > \beta^-_{v_i}, \quad i = 1, \ldots, q. \tag{33}$$

The proof is based on the following claims. First in Claim 1 we show that job h is the only critical job in a schedule with non-adjusted processing times. Then in Claim 2 we demonstrate that such a schedule is not optimal. In Claims 3-4 we show that in an optimal inverse schedule job h remains critical. Finally we give formulae for the required adjustments of the processing times of the jobs and represent them as the constraints of the knapsack problem (28).

We start with the claims about the critical job h.

Claim 1 *In a schedule with non-adjusted processing times, job h is the only critical job.*

Proof of Claim 1. For completeness, define

$$\mathcal{B}_{v_1,v_{i-1}} = 0 \quad \text{if } i = 1.$$

Observe that $a_{v_1} < b_h$ and for any pair of jobs v_i and v_j,

$$a_{v_i} < b_{v_j}.$$

As a consequence,

$$\mathcal{A}_{v_1,v_i} < b_h + \mathcal{B}_{v_1,v_{i-1}}, \quad 1 \le i \le q,$$

so that no job v_i is critical in a schedule with non-adjusted processing time because (6) is violated.

The last two jobs k and ℓ are not critical either:

- for job k, there exists a gap of size T between its completion time on machine A and the starting time on machine B:

$$\begin{aligned} T &= b_h + \mathcal{B}_{v_1,v_q} - \mathcal{A}_{v_1,v_q} - a_k = \left(C + \frac{1}{2} + q\right) + \sum_{i=1}^{q} (w_{v_i} + 1) - \varepsilon q - E = \\ &= \left(C + \frac{1}{2} + q\right) + E - \varepsilon q - E = C + \frac{1}{2} + (1 - \varepsilon)\, q > 0; \end{aligned}$$

- for job ℓ, the gap is even larger: $T + b_k - a_\ell > T$.

Thus for any job, except for job h, a gap between its completion time on machine A and its starting time on machine B is always strictly greater than 0. ∎

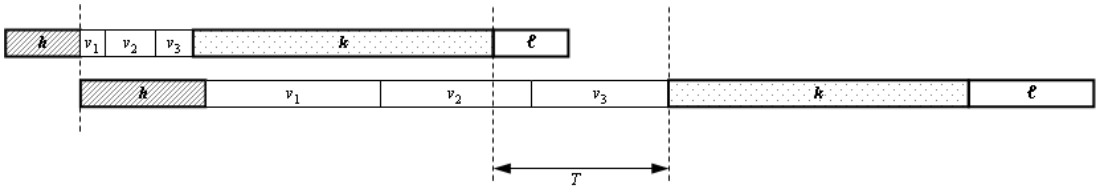

Figure 5: A schedule with non-adjusted processing times

Claim 2 *A schedule with non-adjusted processing times is not optimal.*

Proof of Claim 2. For any job $v_i \in N_1$ that follows the critical job $h \in N_1$, none of the conditions (10) or (11) hold, so that the necessary and sufficient conditions of optimality formulated in Theorem 2 are not satisfied. ■

In what follows we demonstrate that in an optimal solution to the inverse problem job h remains critical.

Claim 3 *No adjustments can result in a new critical v-job.*

Proof of Claim 3. No job v_z, $1 \leq z \leq q$, can become critical since

$$\begin{aligned}
\sum_{i=1}^{z} \widehat{a}_{v_i} &\leq \sum_{i=1}^{z} \overline{a}_{v_i} = (1+\varepsilon)\, z < \\
&< C + (1+\varepsilon)\, z - \varepsilon \leq (C+z) + (z-1)\,\varepsilon < \left(C + \frac{1}{2} + q\right) + (z-1)\,\varepsilon \\
&= b_h + \sum_{i=1}^{z-1} \underline{b}_{v_i} \leq \widehat{b}_h + \sum_{i=1}^{z-1} \widehat{b}_{v_i}
\end{aligned}$$

so that

$$\sum_{i=1}^{z} \widehat{a}_{v_i} < \widehat{b}_h + \sum_{i=1}^{z-1} \widehat{b}_{v_i}.$$

■

Claim 4 *Any adjustments of jobs $\{v_1, \ldots, v_q\}$ cannot make job ℓ critical.*

The proof of the claim follows from the observation that both jobs k and ℓ are non-adjustable, job ℓ is an immediate successor of k and $a_\ell < b_k$.

Claim 5 *If for the adjusted processing times job k becomes critical and h becomes non-critical, then the resulting schedule is not optimal.*

Proof of Claim 5. Job k can become critical. For example, if all v-jobs are fully decompressed on machine A and fully compressed on machine B, then

$$\begin{aligned}
\sum_{i=1}^{q} \overline{a}_{v_i} + a_k &= (1+\varepsilon)\, q + E > (1+\varepsilon)\, q + (C+q)\,, \\
b_h + \sum_{i=1}^{q} \underline{b}_{v_i} &= \left(C + \frac{1}{2} + q\right) + \varepsilon q,
\end{aligned}$$

so that

$$\sum_{i=1}^{q} \overline{a}_{v_i} + a_k > b_h + \sum_{i=1}^{q} \underline{b}_{v_i}.$$

Consider a schedule with adjusted processing times with only one critical job k. In such a schedule, there should be an idle time before the starting time of job k on machine B; otherwise job h is also critical. Such a schedule cannot be optimal since the necessary and sufficient condition of optimality does not hold for the critical job k and the subsequent job ℓ: $k \in N_2$, $\ell \in N_1$. The violated condition for this case is (16) with $h = k$. Observe that processing times of both jobs k and ℓ are fixed and therefore no adjustments can achieve the necessary and sufficient conditions of optimality for the critical job k. ■

Due to the fact that only jobs $v_1, \ldots, v_q$ can be adjusted, Claims 3-5 imply that all possible adjustments resulting in an optimal schedule should keep job h critical and the optimality conditions should be satisfied with respect to this critical job. Since a_{v_i} can only be decompressed and b_{v_i} can only be compressed, the appropriate adjustment for each job v_i is

$$\begin{aligned} \text{either } \widehat{a}_{v_i} &= a_{v_i} + x_{v_i} \text{ with the penalty } \alpha^{+}_{v_i} x_{v_i} \\ \text{or } \widehat{b}_{v_i} &= b_{v_i} - y_{v_i} \text{ with the penalty } \beta^{-}_{v_i} y_{v_i}. \end{aligned}$$

Observe that simultaneous adjustment of both operations of job v_i such that

$$\begin{aligned} \widehat{a}_{v_i} &= a_{v_i} + x_{v_i}, \quad x_{v_i} > 0, \\ \widehat{b}_{v_i} &= b_{v_i} - y_{v_i}, \quad y_{v_i} > 0, \end{aligned}$$

cannot be optimal: a small decompression of $\widehat{b}_{v_i}$ by an amount δ, $0 < \delta < \min\{x_{v_i}, y_{v_i}\}$ and compression of $\widehat{a}_{v_i}$ by the same amount changes the processing times to

$$\begin{aligned} \widehat{\widehat{a}}_{v_i} &= a_{v_i} + x_{v_i} - \delta, \quad x_{v_i} - \delta > 0, \\ \widehat{\widehat{b}}_{v_i} &= b_{v_i} - y_{v_i} + \delta, \quad y_{v_i} - \delta > 0, \end{aligned}$$

and decreases the adjustment cost due to (33), keeping job h critical.

Moreover, for the adjusted processing times necessary and sufficient conditions (10)-(11) should be satisfied with h as a critical job and therefore

$$\begin{aligned} \text{either } \widehat{a}_{v_i} &= a_{v_i} + x_{v_i} = a_h \text{ with the penalty } \alpha^{+}_{v_i}(a_{v_i} - a_h) \text{ (then condition (10) holds);} \\ \text{or } \widehat{b}_{v_i} &= b_{v_i} - y_{v_i} = a_{v_i} \text{ with the penalty } \beta^{-}_{v_i}(b_{v_i} - a_{v_i}) \text{ (then condition (11) holds).} \end{aligned}$$

We introduce $0-1$ variables z_{v_i}, which indicate what type of adjustment is applied to job v_i:

$$z_{v_i} = \begin{cases} 0, & \text{if } \widehat{a}_{v_i} = a_{v_i} + x_{v_i} = a_h, \ \widehat{b}_{v_i} = b_{v_i}, \\ 1, & \text{if } \widehat{a}_{v_i} = a_{v_i}, \qquad\quad \widehat{b}_{v_i} = b_{v_i} - y_{v_i} = a_{v_i}. \end{cases}$$

Then the adjusted processing times can be expressed as

$$\begin{aligned} \widehat{a}_{v_i} &= a_{v_i} z_{v_i} + a_h (1 - z_{v_i}), \\ \widehat{b}_{v_i} &= a_{v_i} z_{v_i} + b_{v_i} (1 - z_{v_i}), \end{aligned} \tag{34}$$

and the corresponding adjustment costs are

$$\alpha_{v_i}^{+}\left(\widehat{a}_{v_i}-a_{v_i}\right)=\alpha_{v_i}^{+}\left(a_h-a_{v_i}\right)\left(1-z_{v_i}\right),$$
$$\beta_{v_i}^{-}\left(b_{v_i}-\widehat{b}_{v_i}\right)=\beta_{v_i}^{-}\left(b_{v_i}-a_{v_i}\right)z_{v_i}.$$

By decompressing the a_{v_i}-values and compressing the b_{v_i}-values job k can become critical. In that case no idle time can appear before the second operation of k (otherwise job k would be the only critical job, a violation of Claim 5):

$$b_h+\sum_{i=1}^{q}\widehat{b}_{v_i}\geq\sum_{i=1}^{q}\widehat{a}_{v_i}+a_k.$$

Substituting expressions (34), we obtain:

$$b_h+\sum_{i=1}^{q}\left[a_{v_i}z_{v_i}+b_{v_i}\left(1-z_{v_i}\right)\right]\geq\sum_{i=1}^{q}\left[a_{v_i}z_{v_i}+a_h\left(1-z_{v_i}\right)\right]+a_k \tag{35}$$

or equivalently

$$\sum_{i=1}^{q}\left(b_{v_i}-a_h\right)z_{v_i}\leq b_h+\sum_{i=1}^{q}b_{v_i}-qa_h-a_k.$$

Thus we can formulate the problem of finding the optimum adjusted processing times as follows:

$$\begin{array}{lll}\min & \sum\limits_{i=1}^{q}\left[\alpha_{v_i}^{+}\left(a_h-a_{v_i}\right)\left(1-z_{v_i}\right)+\beta_{v_i}^{-}\left(b_{v_i}-a_{v_i}\right)z_{v_i}\right] & \\ \text{s.t.} & \sum\limits_{i=1}^{q}\left(b_{v_i}-a_h\right)z_{v_i}\leq b_h+\sum\limits_{i=1}^{q}b_{v_i}-qa_h-a_k, & \\ & z_{v_i}\in\{0,1\}, & 1\leq i\leq q.\end{array}$$

Simplifying the objective function, substituting the values for a_{v_i}, b_{v_i}, a_h, b_h and a_k and using relation (30) we obtain the equivalent formulation:

$$\begin{array}{lll}\max & \sum\limits_{i=1}^{q}\left[\alpha_{v_i}^{+}-\beta_{v_i}^{-}\left(w_{v_i}+1-\varepsilon\right)\right]z_{v_i} & \\ \text{s.t.} & \sum\limits_{i=1}^{q}w_{v_i}z_{v_i}\leq C+\frac{1}{2}-\varepsilon\left(q-\sum\limits_{i=1}^{q}z_{v_i}\right), & \\ & z_{v_i}\in\{0,1\}, & 1\leq i\leq q.\end{array} \tag{36}$$

Since $z_{v_i}\in\{0,1\}$ and due to (31),

$$0\leq\varepsilon\left(q-\sum_{i=1}^{q}z_{v_i}\right)<\frac{1}{2}.$$

The main constraint of (36) can be simplified by using the fact that the sum in the left-hand side and C are integers:

$$\sum_{i=1}^{q}w_{v_i}z_{v_i}\leq C.$$

It is easy to make sure that if the adjustment costs are defined as

$$\alpha^+_{v_i} = \gamma_{v_i} + 1, \qquad 1 \le i \le q,$$
$$\beta^-_{v_i} = \frac{1}{w_{v_i} + 1 - \varepsilon}, \qquad 1 \le i \le q,$$

where γ_{v_i} and w_{v_i} are the parameters of the knapsack problem (28), then problem (36) is equivalent to the knapsack problem (28). ■

Observe that NP-hardness of the inverse counterpart of the permutation flow-shop problem implies NP-hardness of the inverse counterpart of the non-permutation flow-shop problem.

4 Inverse Problem with Adjustable Operations on One Machine

The NP-hardness proof from the previous section relies on the fact that the same job h is critical for the given processing times and for the optimal adjusted processing times, which is justified through a number of claims using the properties of the problem instance. In this section we first demonstrate (via a counterexample) that this is not always the case and that a critical job h of an optimal solution in general cannot be found in advance. Due to this, we introduce n classes of schedules given by a fixed critical job $h = 1, 2, \ldots, n$, see Section 4.1, and look for an optimal solution in each class separately. This approach is illustrated by considering the special case with adjustable operations on one machine. We show how the problem with a fixed critical job h can be solved efficiently. Without loss of generality we assume that the machine which allows adjustments is machine A while processing times on machine B are fixed. In Section 4.2 this case is studied under the assumption that a critical job h belongs to the set $N_1 \cup N_0$; the case of $h \in N_2 \cup N_0$ is studied in Section 4.3.

4.1 Critical Job in an Optimal Solution

In this section we demonstrate that for the inverse problem $F2|\text{adjustable } a_j,\ \pi, \pi|C_{\max}$ with adjustable operations on machine A and fixed operations on machine B, an optimal solution with adjusted processing times may have a critical job, which is not critical for non-adjusted processing times **a** and **b**. Moreover, the necessary and sufficient conditions of optimality should be satisfied for that new critical job.

Consider an instance with the values of a_j and b_j given by Table 3 and only one adjustable operation $\widehat{a}_2$ which processing time can be decreased to $\underline{a}_2 = 1$ and cannot be increased, $\overline{a}_2 = a_2 = 4$. The schedule shown in Fig 6 (a) is based on the non-adjusted processing times and it is not optimal since for $h = 2$ and $v = 4$, $a_h > a_v$ and $a_v < b_v$, a violation of (10) and (11).

As long as $\widehat{a}_2$ is compressed down to a value $\widehat{a}_2 > b_1$, job $h = 2$ is the only critical job and conditions (10) and (11) are violated. If $\widehat{a}_2$ is compressed down to a value $\widehat{a}_2 = b_1$, job 1 also becomes critical, see Fig 6 (b). For the resulting schedule, the optimality conditions (10)-(11) are not satisfied for $h = 2$ because $\widehat{a}_2 > a_4$ and $a_4 < b_4$, but they are satisfied for

j	1	2	3	4
a_j	1	4	5	2
b_j	3	7	3	10

Table 3: Input data of an instance with one adjustable operation $\widehat{a}_2 \in [1, 4]$

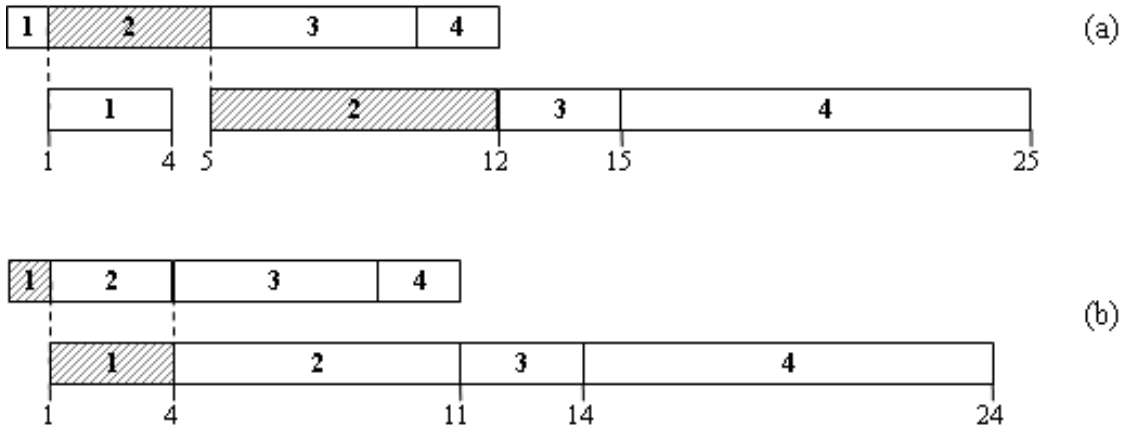

Figure 6: An example with different critical jobs in the initial schedule (a) with non-adjusted processing times and the optimal inverse schedule (b)

$h = 1$:

$$\begin{aligned} a_1 &\leq \widehat{a}_2 < b_2, \\ a_3 &\geq b_3, \\ a_1 &\leq a_4 < b_4. \end{aligned}$$

We conclude that in order to find an optimal schedule, n classes of schedules should be considered with a fixed critical job $h = 1, 2, \ldots, n$, and a global optimum can be found among the solutions determined in these classes.

4.2 Search in the Class of Schedules with $h \in N_1 \cup N_0$

Suppose that machine A allows adjustments while processing times on machine B cannot be changed. We solve the problem $F2|\text{adjustable } a_j,\ \pi, \pi|C_{\max}$ by considering n classes of schedules given by a fixed critical job h, $1 \leq h \leq n$, and finding optimal adjustments in each class; then we select the class of schedules which delivers the smallest possible adjustment cost. In this section we assume that $h \in N_1 \cup N_0$:

$$\widehat{a}_h \leq b_h, \tag{37}$$

and all jobs $u \in \{1, 2, \ldots, h-1\}$ are of type $N_1 \cup N_0$. Then conditions of Theorems 2-3 are applicable. The second case with $h \in N_2 \cup N_0$ and all jobs $v \in \{h+1, \ldots, n\}$ of type $N_2 \cup N_0$ is studied in Section 4.3.

The main idea is to produce a mathematical programming formulation for the selected critical job h using inequalities (6)-(7), (9) and disjunctive inequalities (10)-(11). To handle disjunctive inequalities, we split the interval $[\underline{a}_h, \overline{a}_h]$ of variability of $\widehat{a}_h$ into $O(n)$

subintervals of the form $[t_{k-1}, t_k]$, $1 \leq k \leq q$, where $q \leq 2n$, in such a way that for $\widehat{a}_h \in [t_{k-1}, t_k]$, the type of adjustment of any job $j \in N \setminus \{h\}$ can be uniquely defined as a compression $\widehat{a}_j \leq a_j$ or decompression $\widehat{a}_j \geq a_j$ so that the corresponding term of the objective function $\alpha_j^+ \max\{\widehat{a}_j - a_j, 0\} + \alpha_j^- \max\{a_j - \widehat{a}_j, 0\}$ is linear. In addition, for any job $v \in \{h+1, \ldots, n\}$ disjunctive constraints (10)-(11) can be replaced by a single inequality. This results in a mathematical program with linear constraints and a linear objective function. Having considered q subintervals $[t_{k-1}, t_k]$ in this manner, the solution with the smallest cost provides the solution to the inverse problem.

To produce the mathematical programming formulation which can be solved efficiently by the LP-programming techniques, consider the conditions of Theorems 2-3. Denote the set of jobs processed before h by $U = \{1, 2, \ldots, h-1\}$ and those processed after h by $V = \{h+1, \ldots, n\}$. The corresponding formulation should include

- conditions (6)-(7) which characterize that job h is critical,
- restrictions (9) for each job $u \in U$,
- one of the restrictions (10) or (11) for each job $v \in V$.

Conditions (6)-(7) result in the inequalities of the form:

$$\sum_{j=u}^{h} \widehat{a}_j \geq \mathcal{B}_{u-1,h-1}, \quad 2 \leq u \leq h, \tag{38}$$

$$\sum_{j=h+1}^{v} \widehat{a}_j \leq \mathcal{B}_{h,v-1}, \quad h+1 \leq v \leq n, \tag{39}$$

where the $\mathcal{B}$-values in the right-hand sides are constants.

To model constraints (9) and (10)-(11), we introduce the subintervals $[t_{k-1}, t_k]$ ($k = 1, 2, \ldots, q$) where

$$\underline{a}_h \leq t_0 < t_1 < \cdots < t_q \leq \overline{a}_h$$

are all different values a_j and b_j, $j \in \{1, 2, \ldots, n\}$, contained in $[\underline{a}_h, \overline{a}_h]$. Note that the number of subintervals $[t_{k-1}, t_k]$ is $q \leq 2n$.

Consider such a subinterval $[t_{k-1}, t_k]$, denoted by $[e, f]$, and an adjusted value $\widehat{a}_h \in [e, f]$. Then for any job $u \in U$ one has to satisfy only one of the two inequalities in condition (9):

$$\widehat{a}_u \leq b_u \tag{40}$$

or

$$\widehat{a}_u \leq \widehat{a}_h. \tag{41}$$

The other inequality is satisfied automatically. To demonstrate this, we consider six possible cases of a_u and b_u falling outside (e, f): cases (a)-(c) with $a_u < b_u$ and cases (d)-(f) with $a_u \geq b_u$. Observe that it is assumed that $\widehat{a}_h \in [e, f]$.

(a) If $a_u < b_u \leq e$, then an increased value of $\widehat{a}_u$ is acceptable only if (40) holds. (U_1)

(b) If $a_u \leq e < f \leq b_u$, then an increased value of $\widehat{a}_u$ is acceptable only if (41) holds. (U_2)

(c) If $f \leq a_u < b_u$, then $\widehat{a}_u$ should be decreased to achieve (41) so that (9) is satisfied.(U_3)

(d) If $a_u \geq b_u \geq f$, then $\widehat{a}_u$ should be decreased to achieve (41) so that (9) is satisfied.(U_4)

(e) If $a_u \geq f > e \geq b_u$, then $\widehat{a}_u$ should be decreased to achieve (40) so that (9) is satisfied. (U_5)

(f) If $e \geq a_u \geq b_u$, then $\widehat{a}_u$ should be decreased to achieve (40) so that (9) is satisfied.(U_6)

Observe that in cases (a) and (b) a decreased value of $\widehat{a}_u$, although feasible in terms of (9), cannot be optimal as increasing it back to the initial value a_u keeps inequalities (9) and (38) satisfied but reduces the adjustment cost.

Thus the set of jobs U can be split into subsets U_1, U_2, U_3, U_4, U_5, and U_6, depending on conditions (a), (b), (c), (d), (e) and (f), respectively, and constraints (9) are replaced by (40) for any job $u \in U_1 \cup U_5 \cup U_6$ and by (41) for any job $u \in U_2 \cup U_3 \cup U_4$. The objective function has the following components for the adjustment costs: $\alpha_u^+ (\widehat{a}_u - a_u)$ for $u \in U_1 \cup U_2$ and $\alpha_u^- (a_u - \widehat{a}_u)$ for $u \in U_3 \cup U_4 \cup U_5 \cup U_6$.

Consider now constraints (10)-(11). We show that for any job $v \in V$ they can be reduced to either

$$\widehat{a}_v \geq b_v \tag{42}$$

or

$$\widehat{a}_v \geq \widehat{a}_h. \tag{43}$$

To demonstrate this, we consider six possible cases of a_v and b_v falling outside (e, f): cases (a)-(c) with $a_v < b_v$ and cases (d)-(f) with $a_v \geq b_v$. Observe that it is assumed that $\widehat{a}_h \in [e, f]$.

(a) If $a_v < b_v \leq e$, then a decreased value of $\widehat{a}_v$ cannot lead to condition (10) or (11) satisfied, while $\widehat{a}_v$ increased to the value b_v or higher results in (11) satisfied. Therefore for $\widehat{a}_v$ condition (42) should hold. (V_1)

(b) If $a_v \leq e < f \leq b_v$, then a decreased value of $\widehat{a}_v$ cannot lead to condition (10) or (11) satisfied, while $\widehat{a}_v$ increased to the value $\widehat{a}_h$ results in (10) satisfied; a further increase in $\widehat{a}_v$ may lead to $\widehat{a}_v \geq b_v$ corresponding to (11) satisfied, but it is sufficient to require (43) to satisfy the necessary and sufficient conditions of optimality. (V_2)

(c) If $f \leq a_v < b_v$, then a decreased value of $\widehat{a}_v$ is acceptable only if (43) holds as it is equivalent to (10). (V_3)

(d) If $a_v \geq b_v \geq f$, then a decreased value of $\widehat{a}_v$ beyond b_v is acceptable if (43) holds: for $b_v \leq \widehat{a}_v \leq a_v$ (11) holds, while for smaller values of $\widehat{a}_v$ satisfying (43), condition (10) holds. (V_4)

(e) If $a_v \geq f > e \geq b_v$, then a decreased value of $\widehat{a}_v$ is acceptable only if (42) holds, which is equivalent to (11), and for value of $\widehat{a}_v$ smaller than b_v neither (10) nor (11) hold. (V_5)

(f) If $e \geq a_v \geq b_v$, then a decreased value of $\widehat{a}_v$ is acceptable only if (42) holds, which is equivalent to (11). (V_6)

Observe that in cases (c), (d), (e) and (f) an increased value of $\widehat{a}_v$, although feasible in terms of (10)-(11), cannot be optimal since decreasing it back to the initial value a_v keeps (39) satisfied and at least one of the inequalities (10) or (11) satisfied, but reduces the adjustment cost.

Thus the set of jobs V can be split into subsets V_1, V_2, V_3, V_4, V_5, and V_6, depending on conditions (a), (b), (c), (d), (e) and (f), respectively. Then the disjunctive constraints (10)-(11) can be replaced by (42) for any job $v \in V_1 \cup V_5 \cup V_6$ and by (43) for any job $v \in V_2 \cup V_3 \cup V_4$. The objective function has the following components for the adjustment costs: $\alpha_v^+ (\widehat{a}_v - a_v)$ for $v \in V_1 \cup V_2$ and $\alpha_v^- (a_v - \widehat{a}_v)$ for $v \in V_3 \cup V_4 \cup V_5 \cup V_6$.

Taking into account that the total cost of all adjustments should be as small as possible and that additional constraints $\underline{a}_j \le \widehat{a}_j \le \overline{a}_j$ on lower and upper limits of all variables $\widehat{a}_j$ should be observed, we formulate the corresponding problem as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{u \in U_1 \cup U_2} \alpha_u^+ (\widehat{a}_u - a_u) + \sum_{u \in U_3 \cup U_4 \cup U_5 \cup U_6} \alpha_u^- (a_u - \widehat{a}_u) + \\
& \sum_{v \in V_1 \cup V_2} \alpha_v^+ (\widehat{a}_v - a_v) + \sum_{v \in V_3 \cup V_4 \cup V_5 \cup V_6} \alpha_v^- (a_v - \widehat{a}_v) + \\
& \alpha_h^+ \max\{\widehat{a}_h - a_h, 0\} + \alpha_h^- \max\{a_h - \widehat{a}_h, 0\}
\end{aligned}$$

s.t.

$\widehat{a}_h \le b_h$

$e \le \widehat{a}_h \le f$

$$\left.\begin{aligned}
\widehat{a}_2 + \widehat{a}_3 + \cdots + \widehat{a}_{h-1} + \widehat{a}_h &\ge \mathcal{B}_{1,h-1} \\
\widehat{a}_3 + \cdots + \widehat{a}_{h-1} + \widehat{a}_h &\ge \mathcal{B}_{2,h-1} \\
\ddots &\ \vdots \\
\widehat{a}_{h-2} + \widehat{a}_{h-1} + \widehat{a}_h &\ge \mathcal{B}_{h-3,h-1} \\
\widehat{a}_{h-1} + \widehat{a}_h &\ge \mathcal{B}_{h-2,h-1} \\
\widehat{a}_h &\ge \mathcal{B}_{h-1,h-1}
\end{aligned}\right\} \quad (6')$$

$$\left.\begin{aligned}
\widehat{a}_{h+1} &\le \mathcal{B}_{h,h} \\
\widehat{a}_{h+1} + \widehat{a}_{h+2} &\le \mathcal{B}_{h,h+1} \\
\widehat{a}_{h+1} + \widehat{a}_{h+2} + \widehat{a}_{h+3} &\le \mathcal{B}_{h,h+2} \\
\ddots &\ \vdots \\
\widehat{a}_{h+1} + \widehat{a}_{h+2} + \cdots + \widehat{a}_n &\le \mathcal{B}_{h,n-1}
\end{aligned}\right\} \quad (7')$$

$$\begin{aligned}
&\widehat{a}_u \le b_u, \quad u \in U_1 \cup U_5 \cup U_6 \\
&\widehat{a}_u \le \widehat{a}_h, \quad u \in U_2 \cup U_3 \cup U_4
\end{aligned} \quad (9')$$

$$\widehat{a}_v \ge \widehat{a}_h, \quad v \in V_2 \cup V_3 \cup V_4 \quad (10')$$

$$\widehat{a}_v \ge b_v, \quad v \in V_1 \cup V_5 \cup V_6 \quad (11')$$

$$\underline{a}_j \le \widehat{a}_j \le \overline{a}_j, \qquad j \in N. \quad (44)$$

Observe that one of the components of the objective function $\alpha_h^+ \max\{\widehat{a}_h - a_h, 0\}$ or $\alpha_h^- \max\{a_h - \widehat{a}_h, 0\}$ is zero since for any interval $[e, f] = [t_{k-1}, t_k]$ either $a_h \le e$ or $a_h \ge f$.

We conclude that for $h \in N_1 \cup N_0$ the problem can be handled by solving $q \le 2n$ linear

programming programs, one for each interval $[e, f] = [t_{k-1}, t_k]$. If for a particular interval $[t_{k-1}, t_k]$ no solution exists, then the required adjustment cannot be done with $\widehat{a}_h \in [t_{k-1}, t_k]$ and no solution exists. Observe that it may happen that for the selected critical job h no solution exists for all q intervals $[t_{k-1}, t_k]$.

4.3 Search in the Class of Schedules with $h \in N_2 \cup N_0$

Suppose now that the critical job h is of type $N_2 \cup N_0$:

$$\widehat{a}_h \geq b_h \tag{45}$$

and all jobs from $\{h+1, \ldots, n\}$ are of type $N_2 \cup N_0$. This case represents a symmetric counterpart of the case with $h \in N_1 \cup N_0$ and therefore conditions of Theorems 2-3 should be re-formulated, as described in Section 2.1. To avoid confusion, we denote the jobs which precede h by $K = \{1, 2, \ldots, h-1\}$ and those which follow h by $L = \{h+1, \ldots, n\}$.

The corresponding mathematical programming formulation should include

- conditions (6)-(7) which characterize that job h is critical,
- restrictions (16) on the jobs $\ell \in L$ sequenced after h,
- restrictions (17)-(18) on the jobs $k \in K$ sequenced before h, which should be derived from disjunctive constraints.

Conditions (6)-(7) result in the same inequalities (38)-(39) as in the case of $h \in N_1 \cup N_0$.
Conditions (16) result in the inequalities of the form:

$$b_\ell \leq \widehat{a}_\ell, \tag{46}$$

$$b_\ell \leq b_h, \tag{47}$$

for all $\ell \in L$.

Observe that in (47) both parameters b_ℓ and b_h are constants since processing times of B-operations are non-adjustable. If $b_\ell \leq b_h$ for all $h+1 \leq \ell \leq n$, then there is no need to include these constraints in the mathematical programming formulation; otherwise there does not exist an optimal schedule in the class of the schedule with the selected critical job $h \in N_2 \cup N_0$.

With respect to condition (46), consider the following two cases.

(a) If $a_\ell < b_\ell$, then a decreased value of $\widehat{a}_\ell$ cannot lead to condition (46) satisfied and it should be increased to achieve (46). (L_1)

(b) If $a_\ell \geq b_\ell$, then a decreased value of $\widehat{a}_\ell$ is acceptable only if (46) holds. An increased value cannot be optimal as its compression back to the initial value a_ℓ keeps (7) and (46) satisfied but reduces the adjustment cost. (L_2)

Thus the set of jobs L can be split into subsets L_1 and L_2, depending on conditions (a) and (b), constraint (16) can be replaced by (46) for any job $\ell \in L_1 \cup L_2$, and the objective function has the following components for the adjustment costs: $\alpha_\ell^+ (\widehat{a}_\ell - a_\ell)$ for $\ell \in L_1$ and $\alpha_\ell^- (a_\ell - \widehat{a}_\ell)$ for $\ell \in L_2$.

Consider now conditions (17)-(18) for jobs $k \in K$. We show that they can be either omitted or replaced by

$$b_k \geq \widehat{a}_k. \tag{48}$$

(a) If $b_h \leq b_k$, then there are no constraints on the adjusted value $\widehat{a}_k$. Indeed, if $\widehat{a}_k$ satisfies (48), then condition (18) holds, otherwise condition (17) holds. Observe that a decreased value of $\widehat{a}_k$ cannot be optimal as increasing it back to the initial value a_k keeps (17)-(18) together with (6) satisfied but reduces the adjustment cost; therefore $\widehat{a}_k \geq a_k$ (K_1)

(b) If $b_h > b_k$, then condition (17) cannot hold and therefore the adjusted values of $\widehat{a}_k$ should satisfy (48) corresponding to (18).

(b1) If in addition $b_k > a_k$, then a decreased value of $\widehat{a}_k$ cannot be optimal as increasing it back to the initial value a_k keeps (17)-(18) together with (6) satisfied but reduces the adjustment cost; therefore $\widehat{a}_k$ can only be increased but without violating (48). (K_2)

(b2) If $b_k \leq a_k$, then an increased value of $\widehat{a}_k$ cannot lead to condition (18) satisfied and it should be decreased to achieve (48). (K_3)

Thus the set of jobs K can be split into subsets K_1, K_2 and K_3, depending on conditions (a), (b1) and (b2). Conditions (17)-(18) should be omitted for any $k \in K_1$ and replaced by (48) for any $k \in K_2 \cup K_3$. The objective function should have the following components for adjustment costs: $\alpha_k^+ (\widehat{a}_k - a_k)$ for $k \in K_1 \cup K_2$ and $\alpha_k^- (a_k - \widehat{a}_k)$ for $k \in K_3$.

The corresponding mathematical programming problem can be formulated as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{k \in K_1 \cup K_2} \alpha_k^+ (\widehat{a}_k - a_k) + \sum_{k \in K_3} \alpha_k^- (a_k - \widehat{a}_k) + \\
& \sum_{\ell \in L_1} \alpha_\ell^+ (\widehat{a}_\ell - a_\ell) + \sum_{\ell \in L_2} \alpha_\ell^- (a_\ell - \widehat{a}_\ell) + \\
& \alpha_h^+ \max\{\widehat{a}_h - a_h, 0\} + \alpha_h^- \max\{a_h - \widehat{a}_h, 0\} \\
\text{s.t. } & \widehat{a}_h \geq b_h \\
& \left.\begin{array}{rl}
\widehat{a}_2 + \widehat{a}_3 + \cdots + \widehat{a}_{h-1} + \widehat{a}_h & \geq \mathcal{B}_{1,h-1} \\
\widehat{a}_3 + \cdots + \widehat{a}_{h-1} + \widehat{a}_h & \geq \mathcal{B}_{2,h-1} \\
\ddots & \vdots \\
\widehat{a}_{h-2} + \widehat{a}_{h-1} + \widehat{a}_h & \geq \mathcal{B}_{h-3,h-1} \\
\widehat{a}_{h-1} + \widehat{a}_h & \geq \mathcal{B}_{h-2,h-1} \\
\widehat{a}_h & \geq \mathcal{B}_{h-1,h-1}
\end{array}\right\} \quad (6') \\
& \left.\begin{array}{rl}
\widehat{a}_{h+1} & \leq \mathcal{B}_{h,h} \\
\widehat{a}_{h+1} + \widehat{a}_{h+2} & \leq \mathcal{B}_{h,h+1} \\
\widehat{a}_{h+1} + \widehat{a}_{h+2} + \widehat{a}_{h+3} & \leq \mathcal{B}_{h,h+2} \\
\ddots & \vdots \\
\widehat{a}_{h+1} + \widehat{a}_{h+2} + \cdots + \widehat{a}_n & \leq \mathcal{B}_{h,n-1}
\end{array}\right\} \quad (7') \\
& b_\ell \leq \widehat{a}_\ell, \quad h+1 \leq \ell \leq n, \qquad (16') \\
& b_k \geq \widehat{a}_k, \quad k \in K_2 \cup K_3 \qquad (17')\text{-}(18') \\
& \underline{a}_j \leq \widehat{a}_j \leq \overline{a}_j, \qquad j \in N.
\end{aligned}$$

The above problem can be split into two problems with component $\alpha_h^+ (\widehat{a}_h - a_h)$ of the objective function for $\widehat{a}_h \geq a_h$ and component $\alpha_h^- (a_h - \widehat{a}_h)$ for $\widehat{a}_h \leq a_h$. The resulting two LP problems are similar to problem (44) with two important points of difference:

- they are formulated for two subintervals $[\underline{a}_h, a_h]$ and $[a_h, \overline{a}_h]$ rather than for $O(n)$ subintervals $[t_{k-1}, t_k]$ of $[\underline{a}_h, \overline{a}_h]$;
- each of them can be reduced to the *Resource Allocation Problem with Tree Constraints* (see, e.g., [7, 9]) since unlike the problem from Section 4.2, there are no constraints with variables in both left-hand side and the right-hand side.

Using the results known for the Resource Allocation Problem with Tree Constraints [7, 9], we conclude that the inverse flow shop problem in the case of $h \in N_2 \cup N_0$ can be solved in $O(n \log n)$ time.

5 Conclusions

In this paper we have studied the inverse counterpart of the famous flow shop problem $F2||C_{\max}$. We have produced a new formulation of the necessary and sufficient conditions of optimality which has a number of advantages. First, the new formulation leads to inequality constraints which provide a useful tool for solving the inverse flow shop problem via linear programming. Second, the structure of these inequalities is of a special (tree) type for the permutation flow-shop with adjustable operations on one machine and fixed operations on the other one. Third, unlike earlier research, the conditions are generalized for the non-permutation flow-shop model.

As far as the non-permutation flow shop problem is concerned, the necessary and sufficient conditions of Theorem 4 for the general case are the same as those of Theorems 2-3 for the permutation case; the only difference is related to the inequalities which characterize a critical job. Therefore, for the special case with adjustable operations on one machine, the LP formulations for the permutation and non-permutation case have the same constraints except for those which guarantee that a particular job is critical: conditions (6)-(7) for the permutation case and (20) for the non-permutation case.

Acknowledgement

This research was supported by the EPSRC funded project EP/D059518 "Inverse Optimization in Application to Scheduling". We would like to thank anonymous referees for providing useful recommendations which helped in improving the presentation of the material.

References

[1] Ahuja, R.K., & Orlin, J.B. (2001) Inverse optimization. Operations Research, 49, 771-783.

[2] Brucker, P. (2004) Scheduling Algorithms, Springer, Berlin.

[3] Brucker, P., & Shakhlevich, N.V. (2009) Inverse scheduling with maximum lateness objective. Journal of Scheduling, 12, 475-488.

[4] Duin, C.W., & Volgenant, A. (2006) Some inverse optimization problems under the Hamming distance. European Journal of Operational Research, 170, 887-899.

[5] Gűller, C., & Hamacher, H.W. Capacity inverse minimum cost flow problem. Journal of Combinatorial Optimization (to appear).

[6] Heuberger, C. (2004) Inverse combinatorial optimization: a survey on problems, methods and results, Journal of Combinatorial Optimization, 8, 329-361.

[7] Hochbaum, D.S., & Hong, S.-P. (1995) About strongly polynomial time algorithms for quadratic optimization over submodular constraints, Mathematical Programming, 69, 269 - 309.

[8] Johnson, S.M. (1954) On two and three-stage production scheduling with setup times included. Naval Research Logistics Quarterly 1, 61-67.

[9] Katoh, N., & Ibaraki, T. (1998) Resource allocation problems, in D.-Z. Du and P.M. Pardalos (ed.), Handbook of Combinatorial Optimization, 2, Kluwer, 159-260.

[10] Koulamas, C. (2005) Inverse scheduling with controllable job parameters. International Journal of Services and Operations Management, 1, 35-43.

[11] Lin, Y., & Wang, X. (2007) Necessary and sufficient conditions of optimality for some classical scheduling problems, European Journal of Operational Research, 176, 809-818.

[12] Liu, L.C., & Zhang, J.Z. (2006) Inverse maximum flow problems under the weighted Hamming distance. Journal of Combinatorial Optimization, 12, 395-408.

[13] Tarantola, A. (2005) Inverse Problem Theory and Methods for Model Parameter Estimation, SIAM, Philadelphia.

[14] Wang, L.Z. (2009) Cutting plane algorithms for the inverse mixed integer linear programming problem. Operations Research Letters, 37, 114-116.