ELSEVIER

European Journal of Operational Research 91 (1996) 144–159

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

Variants of the Two Machine Flow Shop Problem connected with factorization of matrix functions

H. Bart, L.G. Kroon *

Erasmus University, P.O. Box 1738, 3000 DR Rotterdam, Netherlands

Received November 1993; revised November 1994

Abstract

In this paper we consider a number of variants of the Two Machine Flow Shop Problem. In these variants the makespan is given and the problem is to find a schedule that meets this makespan, thereby minimizing the infeasibilities of the jobs in a prescribed sense: In the max-variant the maximum infeasibility of the jobs is to be minimized, whereas in the sum-variant the objective is to minimize the sum of the infeasibilities of the jobs. For both variants observations about the structure of the optimal schedules are presented. In particular, it is proved that every instance of these problems has an optimal permutation schedule. It is also shown that the max-variant can be solved by Johnson's Rule. For the sum-variant this is not the case: For solving this problem to optimality something quite different is necessary. Both variants are connected with factorization problems for certain rational matrix functions. The factorizations involved are optimal in some sense and generalize the notion of complete factorization. In this way a connection is established between job scheduling theory on one hand, and mathematical systems theory on the other.

1. Introduction

In this introduction we give a description of the standard Two Machine Flow Shop Problem (2MFSP) and the variants of this problem that are studied in this paper. For more details on 2MFSP we refer to Baker [1].

In 2MFSP there are two machines available for processing n jobs. Every job consists of two operations. However, one of the operations of a job may have zero processing time. The first and second operation of job j are called O_j^1 and O_j^2. The operations O_j^1 and O_j^2 must be processed on the first and the second machine, respectively. Both machines can be processing at most one operation at the same time. Furthermore, processing O_j^2 can not start before processing O_j^1 has been completed.

The processing times of all operations are given and fixed. The processing times of O_j^1 and O_j^2 are denoted by s_j and t_j. Hence an instance J of 2MFSP consists of n tuples (s_j, t_j) specifying the processing times of the operations. Throughout this paper, all processing times are assumed to be integers. This is not a serious restriction. What it amounts to is that all processing times are rationals and that the time unit is chosen appropriately.

* Corresponding author.

SSDI 0377-2217(94)00340-8

If J is an instance of 2MFSP and σ is a feasible schedule for J, then the length of the time interval required for carrying out all jobs is called the *makespan* or the *maximum completion time* of σ and is denoted by $C_{\max}(J, \sigma)$. In 2MFSP the objective is to find a feasible schedule with *minimum* makespan. The minimum makespan is represented by $C^*_{\max}(J)$. Note that Bart and Kroon [4] use the notations $\mu(J, \sigma)$ and $\mu^*(J)$ for the makespan and the minimum makespan of an instance of 2MFSP. In Section 2 we describe some properties of the optimal schedule for an instance of 2MFSP, together with Johnson's Rule that can be used to determine an optimal permutation schedule.

Whereas in standard 2MFSP the objective is to find a feasible schedule with minimum makespan, in the variants of 2MFSP that are studied in this paper a deadline is given and the objective is to find a schedule that meets this deadline, thereby minimizing the infeasibilities of the jobs. Indeed, suppose we have an instance J of 2MFSP with n jobs (s_j, t_j). Furthermore, let the deadline D be an integer satisfying

$$\max\left\{\sum_{j=1}^{n} s_j, \sum_{j=1}^{n} t_j\right\} \leqslant D. \tag{1}$$

In the remainder of this paper we use the notation $S = \sum_{j=1}^{n} s_j$ and $T = \sum_{j=1}^{n} t_j$. In the variants of 2MFSP to be studied it is required that all jobs are completed in D time units. Obviously, if $D < C^*_{\max}(J)$, then this can be achieved only by relaxation of the predecessor constraints. That is, it is allowed that processing O_j^2 already starts before processing O_j^1 has been completed. However, the infeasibilities introduced in this way should be minimized in some sense. This can be made more precise in the following way.

Suppose σ is a schedule for J such that all jobs are completed in D time units. Let $S(O)$ and $F(O)$ denote the start and finish time of operation O, if the processing time of operation O is positive. Furthermore, if $s_j = 0$ then $S(O_j^1) = F(O_j^1) = 0$, and if $t_j = 0$ then $S(O_j^2) = F(O_j^2) = D$. Now the *infeasibility* of job j, denoted by I_j, is defined by

$$I_j = \max\left\{0, F\left(O_j^1\right) - S\left(O_j^2\right)\right\}. \tag{2}$$

Note that $I_j = 0$, if j is a job with $s_j = 0$ or $t_j = 0$. Now we consider the following variants of 2MFSP which differ from each other by their objectives.

2MFSP-M: Minimizing the *maximum infeasibility* of the jobs.

2MFSP-T: Minimizing the *sum of the infeasibilities* of the jobs.

If σ is a schedule for an instance J of 2MFSP-M, then the maximum infeasibility of the jobs is denoted by $\gamma(J, \sigma)$ and the minimum value for $\gamma(J, \sigma)$ is represented by $\gamma^*(J)$. Similarly, if σ is a schedule for an instance J of 2MFSP-T, then the sum of the infeasibilities of the jobs is denoted by $\nu(J, \sigma)$ and the minimum value for $\nu(J, \sigma)$ is written as $\nu^*(J)$.

Note that an operation on the first machine can be pushed backward in time and that an operation on the second machine can be pushed forward in time without increasing the infeasibility of the corresponding job. Therefore we will not consider schedules where the first machine contains operations outside the time interval $(0, S)$ or the second machine contains operations outside the time interval $(D - T, D)$:

Rule 1. *In all considered schedules for an instance of 2MFSP-M or 2MFSP-T the first machine is occupied during the time interval* $(0, S)$ *and the second machine is occupied during the time interval* $(D - T, D)$.

The variants 2MFSP-M and 2MFSP-T of 2MFSP are studied in Sections 3 and 4 of this paper. There it is demonstrated that both problems always have an optimal permutation schedule. Along the way we also deal with some other variants of 2MFSP.

All these variants of 2MFSP bear some analogy to the variant that is studied by Mitten [9]. However, in the latter variant a maximum infeasibility of each job is prescribed and the objective is to find a schedule that minimizes the makespan. On the other hand, in our variants of 2MFSP a deadline is given and the objective is to minimize the infeasibilities of the jobs, thereby taking into account the given deadline. Mitten shows that his

variant can be solved by an extension of Johnson's Rule.

Bart and Kroon [4] show that the mentioned variants of 2MFSP are related to the issue of minimal factorization of rational matrix functions from mathematical systems theory. Roughly speaking, a flow shop problem corresponds to a rational matrix function of a specific type. The correspondence is such that there exists an intimate relationship between the combinatorial aspects of the flow shop problem on one hand and the factorization properties of the rational matrix function on the other. For a brief sketch of the state of affairs and an example, we refer to Section 5 of the present paper.

2. The two machine flow shop problem

It is well known that every instance of 2MFSP has an optimal *non-preemptive* schedule (cf. Baker [1]). That is, once a machine has started processing an operation, it does not start processing another operation until the first operation has been completed. It is also well known that every instance of 2MFSP has an optimal *permutation* schedule. A schedule is a permutation schedule if it is non-preemptive and for all jobs $i \neq j$ with $s_i > 0$, $t_i > 0$, $s_j > 0$ and $t_j > 0$ the operation O_i^1 is processed before the operation O_j^1 on the first machine if and only if the operation O_i^2 is processed before the operation O_j^2 on the second machine. Thus the order of the operations on the first machine is the same as the order of the operations on the second machine.

These properties of 2MFSP can be proved in a straightforward way by applying exchange arguments and by using the fact that, given a feasible schedule, an operation on the first machine can be pushed backward in time without violating the predecessor constraints. Similarly, an operation on the second machine can be pushed forward in time without violating the predecessor constraints.

An optimal permutation schedule for an instance of 2MFSP can be obtained by applying *Johnson's Rule* (cf. Johnson [8]). In Johnson's Rule the list $(p_1, \ldots, p_{2n})$ contains the processing times of al operations in non-decreasing order. This list is called the P(rocessing times)-list. Next, an optimal permutation schedule is created as follows.

- Start with two empty lists. The first list is called the F(irst)-list and the second list is called the L(ast)-list.
- DO WHILE the P-list is non-empty:
  - IF the first number in the P-list equals to s_j for some job j, THEN put job j at the rear of the F-list, ELSE put job j at the front of the L-list.
  - Delete s_j and t_j from the P-list.
- Combine the F-list $(j_1, \ldots, j_p)$ and the L-list $(j_{p+1}, \ldots, j_n)$ into the optimal permutation schedule $(j_1, \ldots, j_p, j_{p+1}, \ldots, j_n)$.

Initially sorting the processing times of the operations can be done in $\mathcal{O}(n \log n)$ time. The remaining part of the algorithm takes $\mathcal{O}(n)$ time. As a consequence, the running time of Johnson's Rule is $\mathcal{O}(n \log n)$. Therefore 2MFSP belongs to the class of easy problems that can be solved in polynomial time (cf. Garey and Johnson [7]).

Example 1. This example considers the instance J of 2MFSP with jobs (3, 4), (4, 6) and (3, 0). Fig. 1 shows the optimal permutation schedule (1, 2, 3) for J. This schedule was obtained by Johnson's Rule. Note that $C^*_{\max}(J) = 13$.

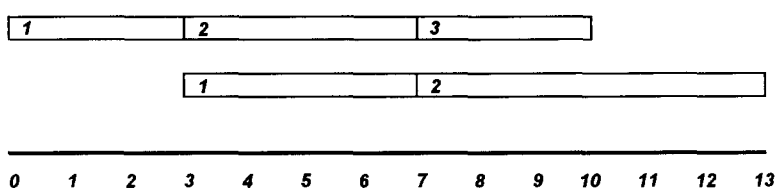

Fig. 1. The optimal permutation schedule (1, 2, 3) for the instance J with jobs (3, 4), (4, 6) and (3, 0).

3. Analysis of the max-variant (2MFSP-M)

In this section we study 2MFSP-M. It is shown that this problem is closely related to 2MFSP. In particular, we will prove Theorem 1, which implies that 2MFSP-M can be solved by Johnson's Rule.

Theorem 1. *If J is an instance of 2MFSP-M, then* $\gamma^*(J) = \max\{C^*_{\max}(J) - D, 0\}$.

Proof. Obviously, $\gamma^*(J) = 0$ if and only if $D \geqslant C^*_{\max}(J)$. Hence in this case we are ready.

Next, we consider the case $D < C^*_{\max}(J)$. Suppose we have an optimal schedule for 2MFSP-M with objective value $\gamma^*(J)$. Then a feasible schedule for 2MFSP is obtained by pushing all operations on the second machine $\gamma^*(J)$ time units forward in time. Thus

$$C^*_{\max}(J) \leqslant D + \gamma^*(J),$$

or

$$\gamma^*(J) \geqslant C^*_{\max}(J) - D.$$

Conversely, suppose we have an optimal schedule for 2MFSP with makespan $C^*_{\max}(J)$. It may be assumed that the first machine is occupied during the time interval $(0, S)$ and that the second machine is occupied during the time interval $(C^*_{\max}(J) - T, C^*_{\max}(J))$. Thus a feasible schedule for 2MFSP-M with objective value not exceeding $C^*_{\max}(J) - D$ is obtained by pushing all operations on the second machine $C^*_{\max}(J) - D$ time units backward in time. As a consequence, $\gamma^*(J) \leqslant C^*_{\max}(J) - D$. □

From the proof of Theorem 1 it is obvious that an optimal schedule for 2MFSP-M corresponds to an optimal schedule for 2MFSP and vice versa. Thus every instance of 2MFSP-M has an optimal permutation schedule that can be obtained by Johnsons's Rule.

Example 2. This example considers the instance J of 2MFSP-M with jobs (3, 4), (4, 6) and (3, 0), and with $D = 10$. This instance was also considered in Example 1. Fig. 2 shows the optimal permutation schedule $\sigma_1 = (1, 2, 3)$ for J. This schedule was obtained by Johnson's Rule. Note that $I_1 = 3$, $I_2 = 3$ and $I_3 = 0$, which gives $\gamma^*(J) = 3$.

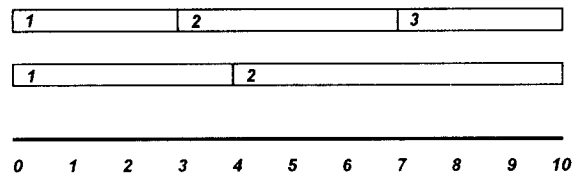

Fig. 2. The optimal permutation schedule (1, 2, 3) for the instance J with jobs (3, 4), (4, 6) and (3, 0), and with $D = 10$.

Corollary 2. *Every instance of 2MFSP-M has an optimal permutation schedule with the property* $S(O_j^1) \leqslant S(O_j^2)$ *and* $F(O_j^1) \leqslant F(O_j^2)$ *for all jobs* j.

Proof. As was noted already, there exists an optimal permutation schedule σ^* that can be obtained by Johnson's Rule. Now let j be a job with $s_j < t_j$. Then, according to Johnson's Rule, all jobs i preceding job j in σ^* have $s_i \leqslant t_i$. Let J_j denote the set of jobs preceding job j in σ^*. Then Rule 1 implies

$$S(O_j^1) = \sum_{i \in J_j} s_i \leqslant \sum_{i \in J_j} t_i \leqslant D - T + \sum_{i \in J_j} t_i = S(O_j^2),$$

$$F(O_j^1) = \sum_{i \in J_j} s_i + s_j \leqslant \sum_{i \in J_j} t_i + t_j \leqslant D - T + \sum_{i \in J_j} t_i + t_j = F(O_j^2).$$

A similar argument holds if $s_j > t_j$. Finally, if $s_j = t_j$, then one of these arguments is valid, depending on whether job j was put at the rear of the F-list or at the front of the L-list in the application of Johnson's Rule. □

4. Analysis of the sum-variant (2MFSP-T)

In this section we study 2MFSP-T. Unfortunately, a similar result as for 2MFSP-M stating

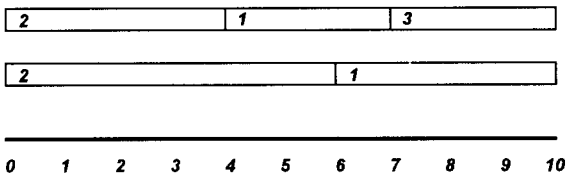

Fig. 3. The optimal permutation schedule (2, 1, 3) for the instance J with jobs (3, 4), (4, 6) and (3, 0), and with $D = 10$.

that an optimal schedule can be found by Johnson's Rule does not hold here.

Example 3. This example considers the instance J of 2MFSP-T with jobs (3, 4), (4, 6) and (3, 0), and with $D = 10$. This instance was also considered in Example 2. Fig. 3 shows the optimal permutation schedule $\sigma_2 = (2, 1, 3)$ for J. Note that $I_1 = 1$, $I_2 = 4$ and $I_3 = 0$, which gives $\nu^*(J) = 5$. Note further that the schedule $\sigma_1 = (1, 2, 3)$ shown in Fig. 2 has $\nu(J, \sigma_1) = 6$.

In spite of the fact that Johnson's Rule does not always produce an optimal schedule for an instance of 2MFSP-T, some information on the structure of an optimal schedule for such an instance can be derived. For example, it will be shown that every instance of 2MFSP-T has an optimal permutation schedule.

This result is obtained by studying two further variants of 2MFSP, which are described now. Let J be an instance of 2MFSP-T, and let σ be a schedule for J. Then an integer time instant $t \in \{0, \ldots, D\}$ is said to be *skipped by job j* if

$$S(O_j^2) < t < F(O_j^1). \tag{3}$$

Furthermore, the integer time instant $t \in \{0, \ldots, D\}$ is said to be *skipped* if it is skipped by at least one job. Note that the time instants 0 and D are not skipped. The total number of skipped time instants is denoted by $\nu_s(J, \sigma)$. In the first further variant of 2MFSP, which is called 2MFSP-S, the objective is to minimize $\nu_s(J, \sigma)$. The minimum value for $\nu_s(J, \sigma)$ is denoted by $\nu_s^*(J)$.

The second further (set of) variant(s) of 2MFSP is called 2MFSP-T(m). This set is indexed by the positive integers m. In 2MFSP-T(m) an infeasibility of a job of $1/m$ time units is not counted. That is, the *m-reduced infeasibility* of job j, which is denoted by $I_j^{(m)}$, is given by

$$I_j^{(m)} = \max\{I_j - 1/m, 0\}$$

$$= \max\{F(O_j^1) - S(O_j^2) - 1/m, 0\}. \tag{4}$$

The sum of the m-reduced infeasibilities of the jobs is denoted by $\nu_{(m)}(J, \sigma)$. In 2MFSP-T(m) the objective is to minimize $\nu_{(m)}(J, \sigma)$. The minimum value is denoted by $\nu_{(m)}^*(J)$.

The variants 2MFSP-S and 2MFSP-T(m) are introduced because they are crucial in the proof that every instance of 2MFSP-T has an optimal permutation schedule. For example, Lemma 4 in the next section is an important auxiliary result in deriving this final result. This lemma holds for 2MFSP-S and 2MFSP-T(m). Unfortunately, it does not hold for 2MFSP-T.

Without loss of generality, Rule 1 is applied to 2MFSP-S and to 2MFSP-T(m). Furthermore, for 2MFSP-S, 2MFSP-5(m) and 2MFSP-T we have the following result.

Lemma 3. *Every instance of 2MFSP-S. 2MFSP-T(m) or 2MFSP-T has an optimal non-preemptive schedule.*

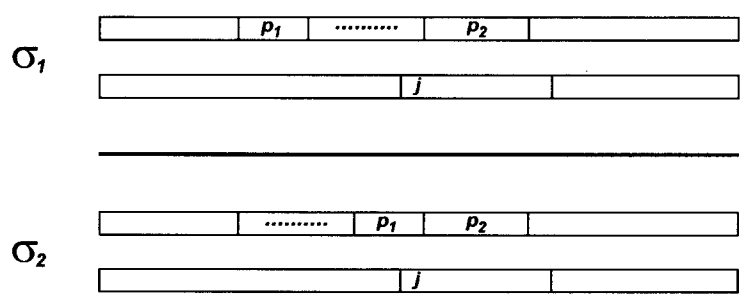

Fig. 4. The transformation from σ_1 to σ_2.

Proof. Suppose we have an optimal preemptive schedule σ_1 where operation O_j^1 is split into two parts p_1 and p_2. Then a new schedule σ_2 is obtained if the parts p_1 and p_2 are combined by pushing p_1 forward in time and by pushing all operations between p_1 and p_2 backward in time. The transformation from σ_1 to σ_2 is illustrated in Fig. 4.

Obviously, the result of this transformation is that the number of preemptions on the first machine decreases, while neither the number of skipped time instants, nor the (reduced) infeasibility of any job increases. A similar transformation decreases the number of preemptions on the second machine. By repeating such transformations as often as necessary, one finally obtains an optimal non-preemptive schedule. □

If σ is a non-preemptive schedule, then Rule 1, together with the fact that all processing times and the deadline D are integers, imply that $S(O)$ and $F(O)$ are integers for all operations O. This implies that for each job j the number of integer time instants t that are skipped by job j equals $I_j^{(1)}$. As a consequence,

$$\nu_s(J, \sigma) \leqslant \sum_{j=1}^{n} I_j^{(1)} = \nu_{(1)}(J, \sigma).$$

Note that strict inequality can occur if a time instant is skipped by two or more jobs.

4.1. Analysis of the skip-variant (2MFSP-S)

In this subsection it is demonstrated that every instance of 2MFSP-S has an optimal permutation schedule. This result is used in subsequent sections to prove a similar result for 2MFSP-T(m) and 2MFSP-T.

Lemma 4. *Every optimal schedule for an instance of 2MFSP-S has the property* $S(O_j^1) \leqslant S(O_j^2)$ *and* $F(O_j^1) \leqslant F(O_j^2)$ *for all jobs* j.

Proof. Suppose we have an optimal schedule σ. Let j be a job with $S(O_j^2) < S(O_j^1)$ in σ. If O_j^1 or O_j^2 has been split into several parts, then the method described in Lemma 3 can be used to transform σ into another optimal schedule σ_1 with $\nu_s(J, \sigma_1) \leqslant \nu_s(J, \sigma)$ where job j is processed in a non-preemptive way. Note that $S(O_j^2) < S(O_j^1)$ in σ_1 as well. All integer time instants t with $S(O_j^2) < t < F(O_j^1)$ are skipped by job j. Further, let the integer time instants τ_0 and τ_1 be defined by:

$$\tau_0 = \max\{t \mid t \leqslant S(O_j^2),\ t \text{ is integer and not skipped}\},$$

$$\tau_1 = \min\{t \mid t \geqslant F(O_j^1),\ t \text{ is integer and not skipped}\}.$$

The time instants τ_0 and τ_1 are well-defined, since 0 and D are not skipped. Furthermore, τ_0 and τ_1 are not skipped, but all integer time instants t with $\tau_0 < t < \tau_1$ are. Note that

$$\tau_0 \leqslant S(O_j^2) < S(O_j^1) < F(O_j^1) \leqslant \tau_1.$$

However, it may happen that $\tau_1 < F(O_j^2)$. Thus let q_j be the processing time of the part of O_j^2

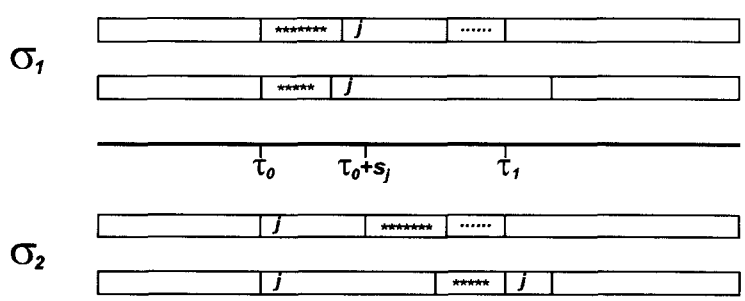

Fig. 5. The transformation from σ_1 to σ_2.

between τ_0 and τ_1 (i.e. $q_j = \min\{t_j, \tau_1 - S(O_j^2)\}$). The cases $s_j \leqslant q_j$ and $s_j > q_j$ are considered separately.

Case 1. $s_j \leqslant q_j$. Since $\tau_0 < \tau_0 + s_j < \tau_1$, the time instant $\tau_0 + s_j$ is skipped in σ_1. Now σ_1 is transformed into a new schedule σ_2 by pushing O_j^1 and the part of O_j^2 between τ_0 and τ_1 backward in time, such that O_j^1 and O_j^2 start at τ_0. At the same time, all parts of operations of other jobs between τ_0 and τ_1 are pushed forward in time as far as is necessary to accomplish this. This transformation is illustrated in Fig. 5.

Now $s_j \leqslant q_j$ implies that the time instant $\tau_0 + s_j$ is not skipped in σ_2. Indeed, suppose job i is such that

$$S(O_i^2) < \tau_0 + s_j < F(O_i^1)$$

in σ_2. Then, obviously, $i \neq j$, which implies

$$S(O_i^2) < \tau_0 < \tau_0 + s_j < F(O_i^1)$$

in σ_2. However, σ_2 differs from σ_1 only between τ_0 and τ_1. Thus

$$S(O_i^2) < \tau_0 < F(O_i^1)$$

in σ_1, which is a contradiction, because the time instant τ_0 is not skipped in σ_1. It can be concluded that the time instant $\tau_0 + s_j$ is not skipped in σ_2.

Case 2. $s_j > q_j$. Note that in this case $q_j = t_j$ and $F(O_j^2) < \tau_1$. Since $\tau_0 < \tau_1 - t_j < \tau_1$, the time instant $\tau_1 - t_j$ is skipped in σ_1. Now σ_1 is transformed into a new schedule σ_2 by pushing O_j^1 and O_j^2 forward in time, such that O_j^1 and O_j^2 finish at τ_1. At the same time, all parts of operations of other jobs between τ_0 and τ_1 are pushed backward in time as far as is necessary to accomplish this. Now it can be shown in a similar way as in Case 1 that the time instant $\tau_1 - t_j$ is not skipped in σ_2.

Hence both in Case 1 and in Case 2 the schedule σ_2 contains an integer time instant t with $\tau_0 < t < \tau_1$ that is not skipped. Since in σ_1 all integer time instants t with $\tau_0 < t < \tau_1$ are skipped, and σ_1 and σ_2 differ from each other only between τ_0 and τ_1, it follows that $\nu_s(J, \sigma_2) < \nu_s(J, \sigma_1)$. Thus an optimal schedule does not contain any job j with $S(O_j^2) < S(O_j^1)$. In a similar way it can be shown that an optimal schedule also does not contain any job j with $F(O_j^2) < F(O_j^1)$. This completes the proof of Lemma 4. □

As we noted already, Lemma 4 is crucial in the proof that every instance of 2MFSP-T has an optimal permutation schedule. Unfortunately, the result of Lemma 4 does not hold for 2MFSP-T directly. The latter is illustrated by the instance of 2MFSP-T with jobs (1, 1) and (1, 1), and with $D = 2$. For this instance all non-preemptive schedules are optimal, but only the permutation schedules satisfy the conclusion of Lemma 4.

Lemma 5. *Every instance of 2MFSP-S has an optimal permutation schedule.*

Proof. Suppose σ_1 is an optimal schedule. It may be assumed that σ_1 is non-preemptive. Furthermore, σ_1 has the property $S(O_j^1) \leqslant S(O_j^2)$ and $F(O_j^1) \leqslant F(O_j^2)$ for all jobs j.

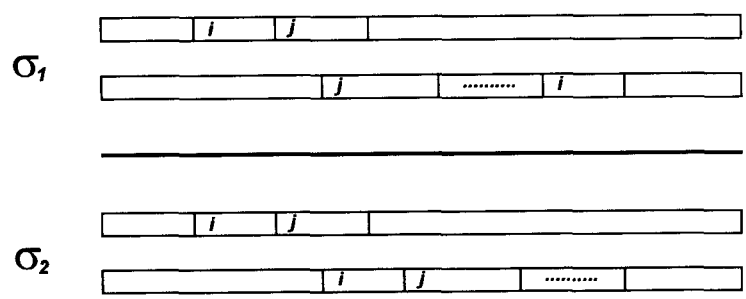

Fig. 6. The transformation from σ_1 to σ_2.

Now let jobs i and j be such that O_i^1 is followed directly by O_j^1 on the first machine, while O_j^2 is followed by O_i^2 on the second machine, possibly with some operations between them. Note that

$$F(O_i^1) = S(O_j^1) \leqslant S(O_j^2) < S(O_i^2).$$

Thus the number of time instants skipped by job i equals zero. Next, a new schedule σ_2 is created by moving operation O_i^2 to the position of O_j^2, while at the same time operation O_j^2, together with all possible operations between O_j^2 and O_i^2 are pushed t_i time units forward in time. The transformation from σ_1 to σ_2 is illustrated in Fig. 6.

In this way the number of time instants skipped by job i remains zero and, obviously, $\nu_s(J, \sigma_2) \leqslant \nu_s(J, \sigma_1)$. Now the order of the operations O_i^1 and O_j^1 on the fist machine is the same as the order of the operations O_i^2 and O_j^2 on the second machine. By applying similar steps as often as necessary, one obtains an optimal permutation schedule. □

4.2. Analysis of the reduced-sum-variant (2MFSP-T(m))

In this subsection it is shown that every instance of 2MFSP-T(m) has an optimal permutation schedule. This result is used in a subsequent subsection to prove a similar result for 2MFSP-T. It is derived by first establishing a connection between 2MFSP-S and 2MFSP-T(1). Indeed, if a non-preemptive schedule σ has the property $S(O_j^1) \leqslant S(O_j^2)$ or $F(O_j^1) \leqslant F(O_j^2)$ for all jobs j, then every skipped time instant t is skipped by exactly one job, as can be seen easily. This result implies $\nu_s(J, \sigma) = \nu_{(1)}(J, \sigma)$ for such schedules σ.

Lemma 6. *A schedule σ^* is an optimal schedule for 2MFSP-S if and only if it is an optimal schedule for 2MFSP-T(1).*

Proof. Suppose σ^* is an optimal schedule for 2MFSP-S. Then $S(O_j^1) \leqslant S(O_j^2)$ and $F(O_j^1) \leqslant F(O_j^2)$ for all jobs j. Thus $\nu_s^*(J) = \nu_s(J, \sigma^*) = \nu_{(1)}(J, \sigma^*)$, as was noted above. Furthermore, for all schedules σ we have the following (in)equalities:

$$\nu_{(1)}(J, \sigma^*) = \nu_s(J, \sigma^*) \leqslant \nu_s(J, \sigma) \leqslant \nu_{(1)}(J, \sigma). \tag{5}$$

As a consequence, σ^* is optimal for 2 MFSP-T(1) as well.

Conversely, suppose σ^* is an optimal schedule for 2MFSP-T(1). Let σ_1^* be an optimal schedule for 2MFSP-S. Then σ_1^* is optimal for 2MFSP-T(1), according to the first part of this proof. Furthermore, σ_1^* has the property $S(O_j^1) \leqslant S(O_j^2)$ and $F(O_j^1) \leqslant F(O_j^2)$ for all jobs j. Thus we have the following (in)equalities:

$$\nu_s(J, \sigma^*) \leqslant \nu_{(1)}(J, \sigma^*) = \nu_{(1)}(J, \sigma_1^*) = \nu_s(J, \sigma_1^*) \tag{6}$$

It follows that σ^* is optimal for 2MFSP-S as well. □

Corollary 7. *Every instance of 2MFSP-T(1) has an optimal permutation schedule. Furthermore, every optimal schedule has the property*

$$S(O_j^1) \leqslant S(O_j^2) \text{ and } F(O_j^1) \leqslant F(O_j^2)$$

for all jobs j.

By using a different time unit $t' = mt$, a similar result can be deduced for 2MFSP-T(m). Thus every instance of 2MFSP-T(m) has an optimal permutation schedule. Also, every optimal schedule has the property $S(O_j^1) \leqslant S(O_j^2)$ and $F(O_j^1) \leqslant F(O_j^2)$ for all jobs j.

4.3. Final analysis of the sum-variant (2MFSP-T)

In this paragraph it is demonstrated that every instance of 2MFSP-T has an optimal permutation schedule. This is achieved by considering 2MFSP-T as the limit of 2MFSP-T(m) in some sense. In particular, we will prove the following lemma.

Lemma 8. *Let J be an instance of 2MFSP-T(m) and 2MFSP-T with n jobs where $m > n$. Then the following holds: If σ^* is an optimal schedule for 2MFSP-T(m), then σ^* is an optimal schedule for 2MFSP-T as well.*

Proof. Suppose σ^* is an optimal schedule for 2MFSP-T(m), but not an optimal schedule for 2MFSP-T. Let σ_1^* be an optimal schedule for 2MFSP-T. Then $\nu(J, \sigma_1^*) \leqslant \nu(J, \sigma^*) - 1$, because $\nu(J, \sigma_1^*)$ and $\nu(J, \sigma^*)$ are integers. This implies the following inequalities:

$$\begin{aligned} \nu_{(m)}(J, \sigma_1^*) &\leqslant \nu(J, \sigma_1^*) \leqslant \nu(J, \sigma^*) - 1 \\ &\leqslant \left(\nu_{(m)}(J, \sigma^*) + n/m\right) - 1 \\ &< \nu_{(m)}(J, \sigma^*). \end{aligned} \tag{7}$$

However, these inequalities imply that σ^* is not optimal for 2MFSP-T(m). This contradiction shows that σ^* is optimal for 2MFSP-T as well. □

Corollary 9. *Every instance of 2MFSP-T has an optimal permutation schedule with the property*

$$S(O_j^1) \leqslant S(O_j^2) \text{ and } F(O_j^1) \leqslant F(O_j^2)$$

for all jobs j.

Note the subtle difference between the Corollaries 7 and 9. Indeed, the earlier mentioned instance of 2MFSP-T with jobs (1, 1) and (1, 1), and with $D = 2$ has an optimal schedule that does not have the property $S(O_j^1) \leqslant S(O_j^2)$ and $F(O_j^1) \leqslant F(O_j^2)$ for all jobs j.

The foregoing reveals a close connection between 2MFSP-T and 2MFSP-T(m). In fact, if J is an instance of 2MFSP-T(m) where $m > n$, then there exists a single permutation schedule σ with $\nu(J, \sigma) = \nu^*(J)$ and $\nu_{(m)}(J, \sigma) = \nu_{(m)}^*(J)$. However, if $m \leqslant n$, then such a permutation schedule need not exist, as is illustrated in Example 4.

Example 4. This example considers the instance J of 2MFSP-T(1) and 2MFSP-T with jobs (2, 3), (2, 3) and (4, 6), and with $D = 12$. Note that two jobs are identical. As a consequence, only three essentially different permutation schedules exist, namely $\sigma_1 = (1, 2, 3)$, $\sigma_2 = (1, 3, 2)$, and $\sigma_3 = (3, 1, 2)$. It is easy to see that $\nu(J, \sigma_1) = 5$, $\nu(J, \sigma_2) = 5$, and $\nu(J, \sigma_3) = 4$. Furthermore, $\nu_{(1)}(J, \sigma_1) = 2$, $\nu_{(1)}(J, \sigma_2) = 3$, and $\nu_{(1)}(J, \sigma_3) = 3$. Thus, $\nu^*(J) = 4$ and $\nu_{(1)}^*(J) = 2$. However, a single permutation schedule σ with $\nu(J, \sigma) = 4$ and $\nu_{(1)}(J, \sigma) = 2$ does not exist. This example will also play a role in the Remark after Lemma 10.

4.4. Finding optimal schedules

Because of the connection of 2MFSP-S, 2MFSP-T(m) and 2MFSP-T with the problem of finding minimal factorizations of certain rational matrix functions (cf. Section 5 and Bart and Kroon [4]), we are interested in optimal instead of approximate schedules for instances of 2MFSP-S, 2MFSP-T(m) and 2MFSP-T. Although the latter problems differ from 2MFSP-M only by their objectives, we have not yet been able to develop polynomial exact algorithms for them. However, it is not difficult to see that one algorithm will be sufficient, because an exact algorithm for any of these problems can be used for all other problems as well. This is expressed in Lemma 10.

Lemma 10. *An exact algorithm for any of the problems 2MFSP-S, 2MFSP-T(m) or 2MFSP-T can be used for all other problems as well.*

Lemma 10 implies that 2MFSP-S, 2MFSP-T(m) and 2MFSP-T have the same computational complexity (cf. Garey and Johnson [7]).

Proof. Lemma 6 shows that an exact algorithm A for 2MFSP-S is also exact for 2MFSP-T(1) and vice versa. Furthermore, by choosing a different time unit $t' = mt$, an exact algorithm A for 2MFSP-T(1) can be used for 2MFSP-T(m) as well.

Next, let A be an exact algorithm for 2MFSP-T(m) and let J be an instance of 2MFSP-T with n jobs. Select a positive integer k such that $km > n$. By choosing a different time unit $t' = kt$, the algorithm A can be used to find an optimal schedule σ for 2MFSP-T(km). Now Lemma 8 implies that σ is optimal for 2MFSP-T as well.

Finally, let A be an exact algorithm for 2MFSP-T and let J be an instance of 2MFSP-T(1)

with deadline D. Then an optimal schedule for 2MFSP-T(1) can be found by applying the algorithm A to the instance J' of 2MFSP-T, where $J' = J \cup \{(0, 1),(1, 0)\}$ with deadline $D + 1$. Indeed, without loss of generality, the additional jobs $(0, 1)$ and $(1, 0)$ of J' are the first and the last jobs in the resulting permutation schedule. □

Remark. We shall now present an explanation why an exact algorithm for 2MFSP-S, 2MFSP-T(m) or 2MFSP-T should be essentially different from Johnson's Rule. First note that Johnson's Rule can be made unambiguous by ordering the processing times of the jobs in the P-list in non-decreasing order, taking into account the following rules:

- If $s_i = t_j$, then s_i precedes t_j in the P-list.
- If $s_i = s_j$, then s_i precedes s_j in the P-list if and only if $i < j$.
- If $t_i = t_j$, then t_i precedes t_j in the P-list if and only if $i < j$.

Hence Johnson's Rule uses information on the ordering of the processing times of the jobs only. Thus if J and J' are instances of 2MFSP with 'isomorphic' P-lists, then there exists a single permutation schedule σ such that $C_{\max}(J, \sigma) = C^*_{\max}(J)$ and $C_{\max}(J', \sigma) = C^*_{\max}(J')$. In particular, the optimal permutation schedule is addition invariant and multiplication invariant That is, the optimal permutation schedule does not change if all processing times are increased with a fixed number of time units or if all processing times are multiplied by a fixed factor.

This is in contrast with 2MFSP-S, 2MFSP-T(m) and 2MFSP-T. Indeed, although an optimal permutation schedule for an instance of 2MFSP-T is multiplication invariant, it need not be addition invariant. The latter is illustrated by the instance J of 2MFSP-T with jobs (1, 3) and (2, 5), and with $D = 8$. For J, the unique optimal permutation schedule is $\sigma_1 = (1, 2)$ with $\nu(J, \sigma_1) = \nu^*(J) = 1$. The permutation schedule $\sigma_2 = (2, 1)$ has $\nu(J, \sigma_2) = 2$. However, if all processing times are increased with 5 time units, then one obtains the instance J' with jobs (6, 8) and (7, 10) and with $D = 18$. For J', the optimal permutation schedule is $\sigma_2 = (2, 1)$ with $\nu(J', \sigma_2) = \nu^*(J') = 10$. The permutation schedule $\sigma_1 = (1, 2)$ has $\nu(J', \sigma_1) = 11$. Similarly, an optimal permutation schedule for an instance of 2MFSP-S or 2MFSP-T(m) need not be addition invariant.

Furthermore, an optimal permutation schedule for an instance of 2MFSP-T(m) need not be multiplication invariant. Indeed, let J be an instance of 2MFSP-T(m) with n jobs. If an optimal permutation schedule for J would be multiplication invariant, then this would imply the existence of a single permutation schedule σ such that $\nu_{(m)}(J, \sigma) = \nu^*_{(m)}(J)$ for all integers m. Furthermore, Lemma 8 would imply that, in particular, $\nu(J, \sigma) = \nu^*(J)$ as well. However, in Example 4 it was shown that for the instance J with jobs (2, 3), (2, 3) and (4, 6), and with $D = 12$, a single permutation schedule σ with $\nu_{(1)}(J, \sigma) = \nu^*_{(1)}(J)$ and $\nu(J, \sigma) = \nu^*(J)$ does not exist. It follows that an optimal permutation schedule for an instance of 2MFSP-T(m) need not be multiplication invariant. Similar remarks are valid for 2MFSP-S as well (cf. Lemma 6).

It can be concluded that an exact algorithm for 2MFSP-S, 2MFSP-T(m) or 2MFSP-T uses more information than just the ordering of the processing times of the jobs, and hence in this sense it has to be essentially different from Johnson's Rule. □

Lemma 11 expresses one more property of 2MFSP-S, 2MFSP-T(m) and 2MFSP-T. This property allows one to decompose an instance into instances of smaller size.

Lemma 11. *Every instance of 2MFSP-S, 2MFSP-T(m) or 2MFSP-T has an optimal permutation schedule with the following property: The list $(j_1, \ldots, j_n)$ representing the optimal permutation schedule consists of three sublists, namely.*

- $(j_1, \ldots, j_{n_1})$ *containing jobs with* $s_j < t_j$,
- $(j_{n_1+1}, \ldots, j_{n_2})$ *containing jobs with* $s_j = t_j$, *and*
- $(j_{n_2+1}, \ldots, j_n)$ *containing jobs with* $s_j > t_j$.

The jobs with $s_j = t_j$ can be arranged in any order.

Proof. Suppose σ^* is an optimal permutation schedule for an instance of 2MFSP-S, 2MFSP-T(m) or 2MFSP-T. Let jobs i and j be consecutive jobs in σ^* with $s_i \geqslant t_i$ and $s_j \leqslant t_j$. If one interchanges the order of i and j, then O_j^2 is pushed backward in time, but O_j^1 is pushed backward in time at least as much, since $s_i \geqslant t_i$. Thus neither the number of time instants skipped by job j nor the (reduced) infeasibility of job j increases. A similar conclusion holds for job i. Furthermore, the other jobs are unaffected by interchanging the jobs i and j. By applying similar interchanges as often as necessary, one obtains an optimal permutation schedule with the desired property. □

According to Lemma 11, jobs j with $s_j = t_j$ do not influence the structure of the optimal schedule. As a consequence, the problem of finding an optimal schedule can be decomposed into two subproblems: an instance J_1 corresponding to the jobs with $s_j < t_j$ and an instance J_2 corresponding to the jobs with $s_j > t_j$. The deadline D_1 to be used in J_1 equals

$$D - \sum_{\{j \mid s_j \geqslant t_j\}} t_j$$

and the deadline D_2 to be used in J_2 equals

$$D - \sum_{\{j \mid s_j \leqslant t_j\}} s_j.$$

Without loss of generality, $s_j < t_j$ for all jobs j in the remaining part of this paper.

In Lemma 12 we show that some instances of 2MFSP-T *can* be solved by sorting the jobs according to non-increasing values of $t_j - s_j$ (cf. the Remark after Lemma 10).

Lemma 12. *If J is an instance of 2MFSP-T with $s_j < t_j$ for all jobs j, $D = T$, and*

$$\sum_{j=1}^{n} (t_j - s_j) \leqslant \min_{j=1,\ldots,n} s_j,$$

then an optimal permutation schedule is obtained by sorting the jobs according to non-increasing values of $t_j - s_j$.

Analogous results hold for 2MFSP-S and 2MFSP-T(m) as well. Note that the condition

$$\sum_{j=1}^{n} (t_j - s_j) \leqslant \min_{j=1,\ldots,n} s_j$$

means that the differences in the processing times are small in comparison with the processing times themselves.

Proof. If σ is a permutation schedule for J, then $D = T$, $s_j < t_j$ for all jobs j, and the inequality $\sum_{j=1}^{n}(t_j - s_j) \leqslant \min_{j=1,\ldots,n} s_j$ imply $S(O_j^2) \leqslant F(O_j^1)$ for all jobs j. Indeed if J_j denotes the set of jobs preceding job j, then

$$\begin{aligned} F(O_j^1) - S(O_j^2) &= \left(\sum_{i \in J_j} s_i + s_j \right) - \sum_{i \in J} t_i = s_j - \sum_{i \in J_j} (t_i - s_i) \\ &\geqslant s_j - \sum_{i=1}^{n} (t_i - s_i) \geqslant 0. \end{aligned}$$

As a consequence, $I_j = F(O_j^1) - S(O_j^2)$ for all jobs j. Now we consider two consecutive jobs i and j where job i precedes job j. We set $A_1 = S(O_i^1)$ and $A_2 = S(O_i^2)$. Then we find

$$\begin{aligned} I_i + I_j &= (A_1 + s_i - A_2) \\ &\quad + (A_1 + s_i + s_j - A_2 - t_i) \\ &= 2(A_1 - A_2) + (s_i + s_j) - (t_i - s_i). \end{aligned}$$

Note that the terms $2(A_1 - A_2)$ and $s_i + s_j$ are independent of the order of the jobs i and j. Hence if $(t_i - s_i) > (t_j - s_j)$, then job i precedes job j in an optimal schedule. If $(t_i - s_i) = (t_j - s_j)$, then jobs i and j can be arranged in any order. □

Some preliminary numerical experiments have revealed that, also in the general case, sorting the jobs according to non-increasing values of $t_j - s_j$ produces schedules that are nearly optimal. This topic is a subject for further research.

As long as we have not been able to find a polynomial algorithm for solving all instances of

2MFSP-S, 2MFSP-T(m) and 2MFSP-T to optimality, we have to be satisfied with an integer program describing these problems. This integer program can be solved by the application of standard branch and bound techniques. Here we will describe the integer program that can be used for solving 2MFSP-T. It bears some similarity to the integer program described by Wagner [10]. The integer program contains the decision variables X_{jp} $(j = 1,\ldots,n;\ p = 1,\ldots,n)$ and I_p $(p = 1,\ldots,n)$ which are defined as follows:

$X_{jp} =$ 1 if job j is carried out at the p-th position of the permutation schedule; 0 otherwise,

$I_p =$ The infeasibility of the job at the p-th position of the permutation schedule.

Note that the decision variables reflect the fact that we can restrict ourselves to permutation schedules. Now the objective and the constraints of the integer program are described as follows.

$$\min \sum_{p=1}^{n} I_p \tag{8}$$

subject to

$$\sum_{p=1}^{n} X_{jp} = 1, \quad j = 1,\ldots,n, \tag{9}$$

$$\sum_{j=1}^{n} X_{jp} = 1, \quad p = 1,\ldots,n, \tag{10}$$

$$\sum_{j=1}^{n} s_j \sum_{q=1}^{p} X_{jq} - \left(D - \sum_{j=1}^{n} t_j \sum_{q=p}^{n} X_{jq} \right) \geqslant I_p, \quad p = 1,\ldots,n, \tag{11}$$

$$I_p \geqslant 0, \quad p = 1,\ldots,n, \tag{12}$$

$$X_{jp} \in \{0, 1\}, \quad j = 1,\ldots,n, \quad p = 1,\ldots,n. \tag{13}$$

The objective (8) specifies that one is interested in minimizing the total infeasibility of the schedule. The assignment constraints (9) and (10) guarantee that every job occupies exactly one position in the schedule and that every position in the schedule is occupied by exactly one job. In the constraints (11) the expression $\sum_{j=1}^{n} s_j \sum_{q=1}^{p} X_{jq}$ represents the completion time of the first operation of the job at the p-th position of the schedule. The expression $D - \sum_{j=1}^{n} t_j \sum_{q=p}^{n} X_{jq}$ represents the start time of the second operation of the job at the p-th position of the schedule. Thus the constraints (11), together with the non-negativity constraints (12), correctly determine the value of the variables I_p.

It is not difficult to modify the above integer program in such a way that it can be used for finding optimal schedules for instances of 2MFSP-S or 2MFSP-T(m). Completing the details is left to the reader.

5. Connection with mathematical systems theory

In this section we provide some background material from mathematical systems theory and we explain briefly how the flow shop problems studied in this paper are related to the issue of minimal factorization of rational matrix functions. For a detailed discussion of this topic we refer to Bart and Kroon [3,4]. For information about the general problem of finding minimal factorizations of rational matrix functions we refer to Bart et al. [2].

Suppose we consider a system S that describes the relation between a k-dimensional input vector $u(t)$ and a k-dimensional output vector $y(t)$ (see Fig. 7). In many cases such an input–output relation can be described by the following system of linear equations:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + u(t), \\ x(0) &= 0. \end{aligned} \tag{14}$$

Here $x(t)$ is an m-dimensional state vector, A is an $m \times m$ matrix, B is an $m \times k$ matrix and C is a $k \times m$ matrix. By considering the Laplace transform of the system (14), one obtains the relation

$$\tilde{y}(\lambda) = W(\lambda)\tilde{u}(\lambda)$$

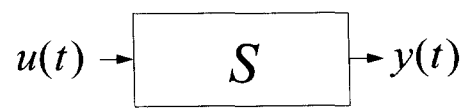

Fig. 7. A linear system S with input $u(t)$ and output $y(t)$.

where

$$W(\lambda) = I_k + C(\lambda I_m - A)^{-1}B. \tag{15}$$

The function W is called the transfer function of the linear system S. Obviously, W is a rational $k \times k$ matrix function with $W(\infty) = I_k$, the k-dimensional unit matrix. As a consequence, W can also be written as

$$W(\lambda) = \left[q_{ij}(\lambda)/p_{ij}(\lambda)\right]_{i,j=1,\ldots,k},$$

where $p_{ij}(\lambda)$ and $q_{ij}(\lambda)$ are complex polynomials with deg p_{ii} = deg q_{ii}, and deg p_{ij} > deg q_{ij} for $i \neq j$ $(i, j = 1, \ldots, k)$.

Conversely, if W is a rational $k \times k$ matrix function with $W(\infty) = I_k$, then it is possible to find matrices A, B and C of appropriate sizes such that (15) is a *realization* of W (cf. Bart et al. [2]). The smallest possible m for which a given function W admits a realization (15) is called the *McMillan degree* of W and is denoted by $\delta(W)$. The realization (15) is called *minimal* if $m = \delta(W)$. If S is a linear system with transfer function W, then the McMillan degree $\delta(W)$ is a measure of the complexity of the system S.

Minimal realizations are essentially unique: If (15) is a minimal realization of W, then all minimal realizations of W can be obtained by replacing A, B and C by MAM^{-1}, MB and CM^{-1} respectively, where M is an invertible $m \times m$ matrix. This result is known as the *state space isomorphism theorem* (cf. Bart et al. [2]).

Two linear systems S_1 and S_2 are *coupled in series* if the output of system S_1 is used as input to system S_2. (see Fig. 8). If S_1 and S_2 are coupled in series and have transfer functions W_1 and W_2, respectively, then $W = W_2W_1$ where W is the transfer function of the system S which is the combination of the linear systems S_1 and S_2.

Conversely, if S is a linear system with transfer function W, and $W = W_2W_1$ is a *factorization* of W, then the original system S can be split up into two linear subsystems S_1 and S_2 which are coupled in series.

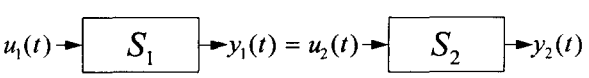

Fig. 8. Linear systems S_1 and S_2 that have been coupled in series.

In general, one is interested in finding linear subsystems with complexities satisfying a certain minimality condition. The latter can be accomplished by looking for linear subsystems whose interaction does not feature redundancies. This corresponds to a so called *minimal factorization*. Hence minimal factorization is an important issue in mathematical systems theory.

We shall now make these things more precise. The McMillan degree $\delta(W)$ has a sublogarithmic property. That is, if $W = W_1 \cdots W_r$ is a factorization of a rational matrix function W into r factors, then

$$\delta(W) \leqslant \delta(W_1) + \cdots + \delta(W_r). \tag{16}$$

A minimal factorization is a factorization with equality in (16). In a minimal factorization pole-zero cancellation does not occur (cf. Bart et al. [2]). There exist non-trivial rational matrix functions that do not allow for any non-trivial minimal factorization. A *complete factorization* of a rational matrix function W is a minimal factorization of W into $\delta(W)$ factors, each with McMillan degree one.

Next, we will explain how the flow shop problems studied in this paper are related to the issue of minimal factorization. To that end, we recall the concept of the *companion based* matrix function (cf. Bart and Kroon [3,4]). A companion based matrix function is a rational $k \times k$ matrix functions admitting a minimal realization (15) where the matrices A and $A^{\times} = A - BC$ are first companion matrices. Here an $m \times m$ matrix M is a *first companion matrix* if it has the form

$$M = \begin{bmatrix} 0 & 1 & & & 0 \\ 0 & 0 & \ddots & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & & & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{m-1} \end{bmatrix}.$$

Bart and Kroon [4] introduced the property of being 'associated' for companion based matrix functions and flow shop problems. Essentially, this property means that a companion based ma-

trix function W is related in a simple way to an instance J of 2MFSP, 2MFSP-M or 2MFSP-T(1), and vice versa. The point here is that, if W and J are associated, then the factorization properties of W and the combinatorial aspects of J are intimately connected.

Example 5. Let the 2×2 rational matrix function W be defined by

$$W(\lambda) = \begin{bmatrix} 1 & 1/p(\lambda) \\ 0 & q(\lambda)/p(\lambda) \end{bmatrix}. \tag{17}$$

where p and q are monic polynomials with the same positive degree. Then it can be verified that W is a companion based matrix function (cf. Bart and Kroon [3]). It should be noted that there also exist companion based matrix functions with a structure different from (17). Now the polynomials p and q can be written as

$$p(\lambda) = (\lambda - \beta_1)^{t_1} \cdots (\lambda - \beta_n)^{t_n},$$

$$q(\lambda) = (\lambda - \beta_1)^{s_1} \cdots (\lambda - \beta_n)^{s_n},$$

where $\beta_1, \ldots, \beta_n$ are n different complex numbers and $\sum_j s_j = \sum_j t_j$. In this case

$$\delta(W) = \sum_j s_j = \sum_j t_j.$$

The concept of association is such that W is associated with the instance J of 2MFSP with the n jobs (s_j, t_j). Furthermore, W is associated with the instance J of 2MFSP-M or 2MFSP-T(1) with this same set of jobs, and with

$$D = \delta(W) = \sum_j s_j = \sum_j t_j.$$

As was stated already, if W and J are associated, then the factorization properties of W and the combinatorial aspects of J are connected. Details are given in the following three theorems which have been established in [4].

Theorem 13. *Let W be a companion based matrix function, let J be an instance of 2MFSP, and assume W and J are associated. Then W admits complete factorization if and only if*

$$C^*_{\max}(J) \leqslant \delta(W) + 1.$$

Since not all rational matrix functions admit complete factorization, one may be interested in minimal factorizations that are optimal in a more general sense. For example, one may be interested in finding a minimal factorization

$$W = W_1 \cdots W_r, \tag{18}$$

where $\max\{\delta(W_i) \mid i = 1, \ldots, r\}$ is minimum. This problem is called the *Max.Degree Problem*. Note that the number of factors r is not pre-specified. The optimal value of the Max.Degree Problem is noted by $\tilde{\gamma}(W)$.

Theorem 14. *Let W be a companion based matrix function, let J be an instance of 2MFSP-M, and assume W and J are associated. Then*

$$\tilde{\gamma}(W) = \max\{\gamma^*(J), 1\}$$

$$= \max\{C^*_{\max}(J) - \delta(W), 1\}.$$

It follows that $\tilde{\gamma}(W)$ can be determined by the application of Johnson's Rule. Note that Theorem 14 is a generalization of Theorem 13. Indeed, according to Theorem 14, $\tilde{\gamma}(W) = 1$ if and only if $C^*_{\max}(J) - \delta(W) \leqslant 1$.

Otherwise one may be interested in finding a minimal factorization (18) with a maximum number of (non-trivial) factors. This problem is called the *Number Problem*. The optimal value of the Number Problem is denoted by $\tilde{\nu}(W)$.

Theorem 15. *Let W be a companion based matrix function, let J be an instance of 2MFSP-T(1), and assume W and J are associated. Then*

$$\tilde{\nu}(W) + \nu^*_{(1)}(J) = \delta(W).$$

Thus a minimal factorization of W with a maximum number of factors corresponds to a schedule for J with minimum total reduced infeasibility. In particular, $\tilde{\gamma}(W) = \delta(W)$ if and only if $\nu^*_{(1)}(J) = 0$. Hence one moment of reflection shows that Theorem 15 provides another generalization of Theorem 13.

Example 6. We finish with an example illustrating the Theorems 13, 14 and 15. This example also

appears in [4]. Consider the rational matrix function W given by

$$W(\lambda) = \begin{bmatrix} 1 & \dfrac{1}{(\lambda+1)^4(\lambda-1)^6} \\ 0 & \dfrac{\lambda^3}{(\lambda+1)(\lambda-1)^2} \end{bmatrix}.$$

Example 5 demonstrates that W is a companion based matrix function, since W has the form (17) with

$$p(\lambda) = (\lambda+1)^4(\lambda-1)^6\lambda^0,$$

$$q(\lambda) = (\lambda+1)^3(\lambda-1)^4\lambda^3.$$

Thus W is associated with the instance of J of 2MFSP with the jobs (3, 4), (4, 6) and (3, 0). This instance is also considered in the Examples 1, 2 and 3 of this paper. By applying Johnson's Rule, we obtain the optimal permutation schedule (1, 2, 3) which is shown in Fig. 1. Since $C^*_{\max}(J) = 13$ and $\delta(W) = 10$, Theorem 13 implies that W does not admit complete factorization.

If J is considered as an instance of 2MFSP-M with $D = 10$, then $\gamma^*(J) = 3$. The optimal permutation schedule for J is shown in Fig. 2. Theorem 14 implies that $\tilde{\gamma}(W) = 3$. Indeed, W admits the minimal factorization

$$\begin{bmatrix} 1 & \dfrac{r_1(\lambda)}{(\lambda+1)^3} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & \dfrac{c_2}{\lambda+1} \\ 0 & \dfrac{\lambda-1}{\lambda+1} \end{bmatrix}\begin{bmatrix} 1 & \dfrac{r_3(\lambda)}{(\lambda-1)^3} \\ 0 & 1 \end{bmatrix}$$
$$\times\begin{bmatrix} 1 & \dfrac{c_4}{\lambda-1} \\ 0 & \dfrac{\lambda}{\lambda-1} \end{bmatrix}\begin{bmatrix} 1 & \dfrac{c_5}{\lambda-1} \\ 0 & \dfrac{\lambda}{\lambda-1} \end{bmatrix}\begin{bmatrix} 1 & \dfrac{c_6}{\lambda-1} \\ 0 & \dfrac{\lambda}{\lambda-1} \end{bmatrix},$$

where $r_1(\lambda) = -\frac{1}{32}(29\lambda^2 + 68\lambda + 41)$, $c_2 = 4$, $r_3(\lambda) = \frac{1}{32}(29\lambda^2 - 68\lambda + 41)$, $c_4 = -4$, $c_5 = 1$ and $c_6 = -1$. Here the McMillan degree of the first and the third factor equals 3, and the other factors have McMillan degree 1. This minimal factorization is optimal for the Max.Degree Problem.

Further, if J is considered as an instance of 2MFSP-T(1) with $D = 10$, then $\nu^*_{(1)}(J) = 3$. The optimal permutation schedule for J is shown in Fig. 3. Theorem 15 implies that $\tilde{\nu}(W) = 10 - 3 = 7$. Indeed, W admits the minimal factorization with 7 factors

$$\begin{bmatrix} 1 & \dfrac{r(\lambda)}{(\lambda-1)^4} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & \dfrac{c_1}{\lambda-1} \\ 0 & \dfrac{\lambda+1}{\lambda-1} \end{bmatrix}\begin{bmatrix} 1 & \dfrac{c_2}{\lambda-1} \\ 0 & \dfrac{\lambda+1}{\lambda-1} \end{bmatrix}$$
$$\times\begin{bmatrix} 1 & \dfrac{c_3}{\lambda+1} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & \dfrac{c_4}{\lambda+1} \\ 0 & \dfrac{\lambda}{\lambda+1} \end{bmatrix}$$
$$\times\begin{bmatrix} 1 & \dfrac{c_5}{\lambda+1} \\ 0 & \dfrac{\lambda}{\lambda+1} \end{bmatrix}\begin{bmatrix} 1 & \dfrac{c_6}{\lambda+1} \\ 0 & \dfrac{\lambda}{\lambda+1} \end{bmatrix},$$

where $r(\lambda) = \frac{1}{32}(99\lambda^3 - 345\lambda^2 + 411\lambda - 169)$, $c_1 = \frac{699}{64}$, $c_2 = -\frac{381}{16}$, $c_3 = -\frac{1}{64}$, $c_4 = 12$, $c_5 = 3$ and $c_6 = 1$. This minimal factorization is optimal for the Number Problem.

In Examples 5 and 6 we started with a companion based matrix function and we constructed an instance of 2MFSP, 2MFSP-M or 2MFSP-T(1) associated with it. From this construction it is clear that it is also possible to go in the reverse direction. In fact, it is clear how, given an instance J with jobs (s_j, t_j) satisfying $\Sigma_j s_j = \Sigma_j t_j$, one can produce a companion based matrix function W associated with J. The requirement $\Sigma_j s_j = \Sigma_j t_j$ is not a serious restriction, since it can always be met by the addition of a dummy job. The conclusion is that the flow shop problems considered in this paper and the corresponding factorization problems are equivalent to a certain extent.

References

[1] Baker, K.R., *Introduction to Sequencing and Scheduling*, Wiley, New York, 1975.

[2] Bart, H., Gohberg, I., and Kaashoek, M.A., "Minimal factorization of matrix and operator functions", in: *Oper-*

ator Theory: Advances and Applications, Vol 1., Birkhäuser, Basel, 1979.

[3] Bart, H., and Kroon, L.G., "Companion based matrix functions: Description and minimal factorization", Management Report Series 133, Rotterdam School of Management, Erasmus University Rotterdam, 1993. Forthcoming in *Linear Algebra and its Applications*.

[4] Bart, H., and Kroon, L.G., "Factorization and job scheduling: A connection via companion based matrix functions", Management Report Series 178, Rotterdam School of Management, Erasmus University Rotterdam, 1994. Forthcoming in *Linear Algebra and its Applications*.

[5] Conway, R.W., Maxwell, W.L., and Miller, L.W., *Theory of Scheduling*, Addison-Wesley, Boston, MA, 1967.

[6] Dudek, R.A., Panwalkar, S.S., and Smith, M.L., "The lessons of flowshop scheduling research", *Operations Research* 40 (1992) 7–13.

[7] Garey, M.R., and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

[8] Johnson, S.M., "Optimal two- and three stage production schedules with setup times included", *Naval Research Logistics Quarterly* 1/1 (1954).

[9] Mitten, L.G., "Sequencing n jobs on two machines with arbitrary time lags", *Management Science* 5/3 (1959).

[10] Wagner, H.M., "An integer programming model for machine scheduling", *Naval Research Logistics Quarterly* 6 (1959) 131–140.