ELSEVIER

Discrete Applied Mathematics 112 (2001) 199–216

DISCRETE APPLIED MATHEMATICS

Makespan minimization for flow-shop problems with transportation times and a single robot ☆

Johann Hurink[a], Sigrid Knust[b,*]

[a] *University of Twente, Faculty of Mathematical Sciences, NL-7500 AE Enschede, Netherlands*
[b] *Universität Osnabrück Fachbereich Mathematik/Informatik, D-49069 Osnabrück, Germany*

Received 1 April 1998; revised 24 March 1999; accepted 1 August 2000

Abstract

In a flow-shop problem with transportation times and a single robot n jobs consisting of m operations have to be processed in the same order on m machines. Additionally, transportation times are considered if a job changes from one machine to another. We assume that unlimited buffer space exists between the machines and all transportations have to be done by a single robot. The objective is to determine a feasible schedule with minimal makespan.

New complexity results are derived for special cases where the processing or transportation times are constant values. Some of these may also be interpreted as new results for special cases of the classical 3-machine flow-shop $F3||C_{\max}$ with constant processing times at certain stages.

Keywords: Flow shop; Transportation times; Robot; Complexity results

1. Introduction

A flow-shop problem with transportation times and a single robot is a generalization of the classical flow-shop problem and may be formulated as follows:

We are given m machines $M_1,\ldots,M_m$ and n jobs $j=1,\ldots,n$. Each job j consists of m operations O_{ij} $(i=1,\ldots,m)$ which have to be processed in the order $O_{1j}\rightarrow O_{2j}\rightarrow\cdots\rightarrow O_{mj}$. Operation O_{ij} has to be processed on machine M_i without preemption for $p_{ij}\geqslant 0$ time units. Each machine can only process one job at a time.

Additionally, transportation times are considered. They occur if a job changes from one machine to another, i.e. if operation O_{kj} is processed on machine M_k and afterwards $O_{k+1,j}$ on machine M_{k+1}, a transportation time $t_{jk}\geqslant 0$ arises. These transportation times

☆ Supported by the Deutsche Forschungsgemeinschaft, Project 'Komplexe Maschinen-Schedulingprobleme'.
* Corresponding author.
E-mail address: sigrid@mathematik.uni-osnabrueck.de (S. Knust).

0166-218X/01/$ - see front matter
PII: S0166-218X(00)00316-4

may be job-dependent or job-independent ($t_{jk} = t_k$). We consider the case where all transportations are done by a single transport robot R which can only handle one job at a time. Thus, conflicts between transportations may arise and a job may have to wait for the robot before its transportation. We assume that empty moving times of the robot are negligible, i.e. if the robot moves empty without carrying a job no additional times are considered. All values p_{ij} and t_{jk} are supposed to be non-negative integers.

We assume that we have unlimited buffer space between the machines. Each machine M_k has an unlimited output buffer where jobs processed on M_k and waiting for the robot may be stored. The jobs are automatically transferred into this buffer and no further times for this transfer are considered. Additionally, each machine M_k has an unlimited input buffer where jobs awaiting processing on M_k may be stored.

The objective is to determine a feasible schedule which minimizes the makespan $C_{\max} = \max_{j=1}^{n} C_j$, where C_j is the finishing time of the last operation O_{mj} of job j.

To describe special cases of this problem we extend the $\alpha|\beta|\gamma$-notation of Graham et al. [5]. In the α-field we write $R1$ to indicate that one robot is available for all transportations. In the β-field we add information about the transportation times: t_{jk} means that the transportation times are job- and machine-dependent, $t_{jk} = t_k$ denotes machine-dependent but job-independent transportation times. In practice this situation may occur for jobs of the same size where the transportation times only depend on the travel distances between the machines. $t_{jk} = T$ indicates that all transportation times are equal to a constant T. If we have only $m=2$ machines, the robot always transports from M_1 to M_2. Therefore, the index k in the notation t_{jk} is dropped and the transportation times are denoted by t_j. If they may take only two values T_1, T_2, we write $t_j \in \{T_1, T_2\}$ in the β-field. In practice this situation may occur if we are given two types of jobs, some "big" and some "small" jobs where it takes longer to transport the big jobs than the smaller ones.

Throughout the paper we will also consider the special cases where all operations O_{ij} have unit processing times $p_{ij} = 1$ or constant processing times $p_{ij} = p$. We have to distinguish these cases since, due to the integrality of the data, these cases are different from the complexity point of view. For simplifying the presentation, in the second case we will also normalize the data in such a way that $p_{ij} = 1$ holds and allow the transportation times to be rational numbers.

In recent years various models have been proposed where the interactions between classical job scheduling decisions and transportation aspects are considered:

- Many authors concentrated on cyclic flow-shop scheduling problems for robotic cells without any buffers (for an overview see Crama et al. [2]). In most cases repetitive manufacturing is studied where the long-run average cycle time has to be minimized. For special situations polynomial algorithms have been obtained by Sethi et al. [14], Crama and van de Klundert [1], and Hall et al. [6]. Robotic cells with a single unit of buffer behind each machine have been studied by Finke et al. [3].
- Considering the case in which sufficient robots are available for transportation leads to problems where the transportation times only correspond to minimal time-lags (delays) between operations of the same job. In this case the transportation stage

becomes a non-bottleneck one and there are no conflicts between jobs awaiting transportation. In this situation Johnson's algorithm for problem $F2||C_{\max}$ [7] can be used to solve problem $F2|t_j{=}T|C_{\max}$ as well as the permutation variant of $F2|t_j|C_{\max}$ (Mitten [12]). The non-permutation variant is much harder, since Yu [16] has shown that problem $F2|\,p_{ij}=1,t_j|C_{\max}$ is already strongly $\mathcal{NP}$-hard. Additionally, he showed that $F2|t_j\in\{T_1,T_2\}|C_{\max}$ is strongly $\mathcal{NP}$-hard and $F2|\,p_{ij}=1,t_j\in\{T_1,T_2\}|C_{\max}$ is polynomially solvable.

- For flow-shop problems with transportation times, limited buffer space and a single robot, some polynomially solvable special cases have been described by Panwalkar [13] and Levner et al. [11]. Kise et al. [9] and Stern and Vitner [15] considered the two-machine case with an additional no-wait condition which can be formulated as a specially structured traveling salesman problem.
- For the flow-shop environment with a single robot and unlimited buffer space only one complexity result is available. Kise [8] has shown that problem $F2,R1|t_j{=}T|C_{\max}$ is $\mathcal{NP}$-hard in the ordinary sense.

This paper is organized as follows. In Section 2 we consider problems with two machines and strengthen the result of Kise [8] by showing that $F2,R1|t_j=T|C_{\max}$ is strongly $\mathcal{NP}$-hard. Furthermore, we show that $F2,R1|\,p_{ij}=p,t_j|C_{\max}$ is $\mathcal{NP}$-hard in the strong sense and that $F2,R1|\,p_{ij}=p,t_j\in\{T_1,T_2\}|C_{\max}$ and $F2,R1|\,p_{ij}=1,t_j|C_{\max}$ are polynomially solvable. In Section 3 we consider the problem $F,R1|n\geqslant m-1,\ p_{ij}=p,t_{jk}=t_k|C_{\max}$ with m machines and show that it is polynomially solvable. The paper ends with some concluding remarks.

2. Flow-shop problems with two machines

In this section we consider problems in which we have two machines M_1,M_2, one robot R and n jobs j with processing times p_{1j} on M_1 and p_{2j} on M_2. Furthermore, for each job j a transportation time t_j from M_1 to M_2 is given. This problem, $F2,\ R1|t_j|C_{\max}$, is equivalent to the classical three-machine flow-shop $F3||C_{\max}$ where the second machine corresponds to the robot R.

In the Introduction we have stated that we do not consider empty moving times. It is easy to see that for $m=2$ machines the more general situation with empty moving times $t'_{21}\neq 0$ (arising for the robot when it moves back from machine M_2 to M_1 after having transported a job) is equivalent to our model without empty moving times. Given a problem with $t'_{21}\neq 0$, we construct an instance of the corresponding problem with $t'_{21}=0$ by adding t'_{21} to all transportation times t_j. Each feasible schedule for the resulting problem can be transformed into a feasible schedule for the original problem and vice versa: we shift the whole schedule on machine M_2 by t'_{21} time units to the left (right), which changes the objective value by only a constant.

For problem $F2,R1|t_j|C_{\max}$ we may restrict the search for an optimal solution to permutation plans, since for problem $F3||C_{\max}$ an optimal permutation plan always exists. The objective value of a permutation π is given by the value of a critical path.

Such a path contains at first some operations on the first machine, then on the robot and finally on the second machine:

$$F(\pi) = \max_{k=1}^{n} \max_{l=k}^{n} \left\{ \sum_{\lambda=1}^{k} p_{1\pi_\lambda} + \sum_{\lambda=k}^{l} t_{\pi_\lambda} + \sum_{\lambda=l}^{n} p_{2\pi_\lambda} \right\}. \tag{2.1}$$

In the following sections we consider two special cases of this problem and show that they are both $\mathcal{NP}$-hard in the strong sense. In the first case we have arbitrary processing times of the jobs, but constant transportation times; in the second we have arbitrary transportation times, but constant processing times of all operations. These two cases correspond to the classcical 3-machine problems $F3|p_{2j} = p_2|C_{\max}$ and $F3|p_{1j} = p_{3j} = p|C_{\max}$. In Sections 2.3 and 2.4 we consider special cases of problem $F2, R1|p_{ij} = p, t_j|C_{\max}$ which are polynomially solvable. In the first case we have only two transportation time values T_1 and T_2, and in the second case we assume unit processing times $p_{ij} = 1$.

2.1. Problem $F2, R1|t_j = T|C_{\max}$

In problem $F2, R1|t_j = T|C_{\max}$ we have arbitrary processing times on M_1 and M_2, but all transportation times t_j are equal to a constant T. The equivalent version of this problem with empty moving times t'_{21} has been shown to be $\mathcal{NP}$-hard in the ordinary sense by Kise [8]. In the following we strengthen Kise's result by showing that the problem is strongly $\mathcal{NP}$-hard. Within the proof of the result we will use the following reformulation of the objective value (2.1) for a permutation π in the considered special case of equal transportation times:

$$F(\pi) = \max_{k=1}^{n} \max_{l=k}^{n} \left\{ \sum_{\lambda=1}^{k} p_{1\pi_\lambda} + (l-k+1)T + \sum_{\lambda=l}^{n} p_{2\pi_\lambda} \right\}. \tag{2.2}$$

Theorem 1. *Problem $F2, R1|t_j = T|C_{\max}$ is $\mathcal{NP}$-hard in the strong sense.*

Proof. We will show that the strongly $\mathcal{NP}$-complete problem 3-PARTITION is reducible to the decision problem of $F2, R1|t_j = T|C_{\max}$.

3-PARTITION (3-PART): Given $3m$ positive numbers $a_1, \ldots, a_{3m}$ with $\sum_{i=1}^{3m} a_i = mb$ and $b/4 < a_i < b/2$ for $i = 1, \ldots, 3m$, does there exist a partition $I_1, \ldots, I_m$ of the index set $\{1, \ldots, 3m\}$ such that $|I_j| = 3$ and $\sum_{i \in I_j} a_i = b$ for $j = 1, \ldots, m$?

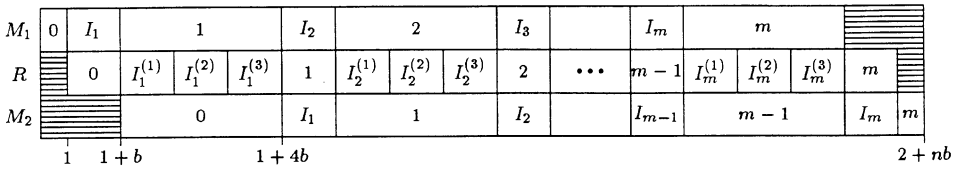
W.l.o.g. we can assume that in the instance of 3-PART $a_i > 1$ holds for all i (if this is not the case, we scale the data in an appropriate way). Given an arbitrary instance of 3-PART, we construct the following instance of the flow-shop problem $F2, R1|t_j = T|C_{\max}$ with $n := 4m + 1$ jobs:

job 0 :	$p_{10} = 1,$	$p_{20} = 3b,$	
job m :	$p_{1m} = 3b,$	$p_{2m} = 1,$	
jobs $1, \ldots, m-1$:	$p_{1j} = 3b,$	$p_{2j} = 3b$	for $j = 1, \ldots, m-1,$
jobs $m+1, \ldots, 4m$:	$p_{1,i+m} = a_i,$	$p_{2,i+m} = a_i$	for $i = 1, \ldots, 3m.$

The transportation time T is set to $T = b$ and we ask for a permutation π with $F(\pi) \leqslant y := 2 + (4m + 1)b = 2 + nb$.

We show that a permutation π for the constructed instance with $F(\pi) \leqslant y$ exists if and only if 3-PART has a solution.

Assume that $I_1, \ldots, I_m$ is a solution of 3-PART. Then we define a permutation π as follows:

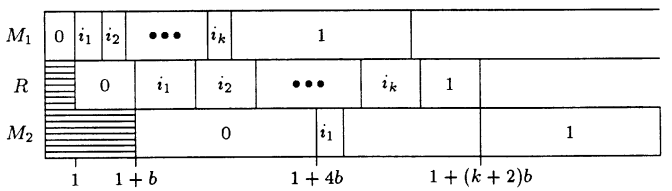


Obviously, this permutation π fulfills $F(\pi) \leqslant y$.

Conversely, assume that the flow-shop problem has a solution π with $F(\pi) \leqslant y$. By setting $k = 1$, $l = n$ in (2.2) we get for all permutations π:

$$F(\pi) \geqslant p_{1\pi_1} + nT + p_{2\pi_n} \geqslant 1 + nb + 1 = y.$$

Thus, for a permutation π with $F(\pi) = y$ we may conclude that
(1) job 0 is processed at the first position, since $p_{1i} > 1$ for $i \neq 0$,
(2) job m is processed at the last position, since $p_{2i} > 1$ for $i \neq m$,
(3) machine M_1 processes jobs in the interval $[0, 1 + (n-1)b]$ without idle time,
(4) robot R transports jobs in the interval $[1, 1 + nb]$ without idle time,
(5) machine M_2 processes jobs in the interval $[1 + b, 2 + nb]$ without idle time.
W.l.o.g. we can assume that the jobs $1, \ldots, m-1$ are processed w.r.t. increasing numbers. Let $I_1 := \{i_1, \ldots, i_k\}$ be the set of jobs scheduled between jobs 0 and 1. We have $I_1 \neq \emptyset$, since otherwise there would be an idle period on the robot between jobs 0 and 1, which contradicts (4).



In the following we will show that $k = 3$ and $\sum_{i \in I_1} a_i = b$ hold. We use the variables C_{ij}^{π} denoting the completion time of job j on machine M_i in the permutation π. The values of these variables for the jobs in the set I_1 are given by

$$C_{1i_\mu}^{\pi} = 1 + \sum_{\lambda=1}^{\mu} p_{1i_\lambda} < 1 + \mu \cdot \frac{b}{2} < 1 + \mu b \quad (\mu = 1, \ldots, k),$$

i.e. the robot transports jobs $0, i_1, \ldots, i_k$ without idle time in the interval $[1, 1+(k+1)b]$.

- If $k \leqslant 2$ holds, we have $C_{11}^{\pi} = 1 + \sum_{\lambda=1}^{k} p_{1i_\lambda} + 3b > 1 + 3b$ and the robot finishes the transportation of job i_k at time $1 + (k+1)b$ and, thus, not after time $1 + 3b$. Hence, the robot has an idle period between jobs i_k and 1, which contradicts (4).
- If $k \geqslant 4$ holds, we have

$$\sum_{\lambda=1}^{k} p_{2i_\lambda} < k \cdot \frac{b}{2} \leq k \cdot \frac{b}{2} + (k-4) \cdot \frac{b}{2} = (k-2)b.$$

On the other hand, job 1 cannot start on M_2 earlier than time $1 + (k+2)b$, since jobs 0,1 and all jobs of I_1 have to be transported before. Thus, the time period between the completion time $C_{20}^{\pi} = 1 + 4b$ of job 0 on M_2 and the starting time of job 1 on M_2 is not completely filled with jobs from I_1, which contradicts (5).

Thus, we must have $k = 3$. This implies that job 1 is transported by the robot in the interval $[1 + 4b, 1 + 5b]$, i.e. job 1 starts on M_1 not after time $1 + b$.

Therefore, $\sum_{i \in I_1} p_{1i} \leqslant b$. On the other hand, job 1 starts on M_2 not before $1 + 5b$. Since we have no idle time on M_2 in $[1 + b, 2 + nb]$, we must have $4b + 1 + \sum_{\lambda=1}^{3} p_{2i_\lambda} \geqslant 1 + 5b$. Therefore, $\sum_{i \in I_1} p_{2i} \geqslant b$. Since $p_{1i} = p_{2i} = a_i$, we have $\sum_{i \in I_1} a_i = b$.

Analogously, we show that the remaining sets $I_2, \ldots, I_m$ separated by the jobs $1, \ldots, m$ contain 3 elements and fulfill $\sum_{i \in I_j} a_i = b$ for $j = 2, \ldots, m$. Thus, $I_1, \ldots, I_m$ define a solution of 3-PART. □

2.2. Problem $F2, R1 | p_{ij} = p, t_j | C_{\max}$

In problem $F2, R1 | p_{ij} = p, t_j | C_{\max}$ we have constant processing times of all operations on M_1 and M_2, but arbitrary transportation times between the machines. For the sake of simplicity we normalize the data in such a way that we have unit processing times on M_1 and M_2, but rational, job-dependent transportation times $t_j \geqslant 0$ from M_1 to M_2. Now, the objective value (2.1) of a permutation π reduces to

$$\begin{aligned} F(\pi) &= \max_{k=1}^{n} \max_{l=k}^{n} \left\{ k + \sum_{\lambda=k}^{l} t_{\pi_\lambda} + (n - l + 1) \right\} \\ &= \max_{k=1}^{n} \max_{l=k}^{n} \left\{ \sum_{\lambda=k}^{l} (t_{\pi_\lambda} - 1) \right\} + n + 2. \end{aligned}$$

If $\min_{j=1}^{n} \{t_j\} \geqslant 1$ holds, the transportation times dominate the processing times and we have $F(\pi) = \sum_{j=1}^{n} (t_j - 1) + n + 2$ for each permutation π. If $\max_{j=1}^{n} \{t_j\} \leqslant 1$ holds, the transportation times are dominated by the processing times and we have $F(\pi) = \max_{j=1}^{n} \{t_j - 1\} + n + 2$ for each permutation π. Since both values are a lower bound for the problem, each permutation is optimal in these cases.

Defining $s_j := t_j - 1$, the decision version of our problem corresponds to the following problem:

P1: Given are n rational numbers $s_1, \ldots, s_n \geqslant -1$ and a threshold value y. Does there exist a permutation π with $\tilde{F}(\pi) = \max_{k=1}^{n} \max_{l=k}^{n} \{\sum_{\lambda=k}^{l} s_{\pi_\lambda}\} \leqslant y$?

We will prove that our problem is $\mathcal{NP}$-hard by showing that P1 is $\mathcal{NP}$-complete.

Theorem 2. *Problem* $F2,R1|p_{ij}=p,t_j|C_{\max}$ *is* $\mathcal{NP}$*-hard in the strong sense.*

Proof. We will show again that the strongly $\mathcal{NP}$-complete problem 3-PARTITION is reducible to problem P1.

W.l.o.g. we can assume that in the instance of 3-PART $b<1$ holds (otherwise we scale the data in an appropriate way). Given an arbitrary instance of 3-PART, we construct the following instance of P1 with $n:=4m-1$ integers:

$$s_i:=a_i \text{ for } i=1,\dots,3m,$$

$$s_j:=-b \text{ for } j=3m+1,\dots,4m-1.$$

We ask for a permutation π with $\tilde{F}(\pi)\leqslant y:=b$.

Assume that $I_1,\dots,I_m$ is a solution of 3-PART. Then we define

$$\pi:=(\underset{b}{I_1},\ \underset{-b}{3m+1},\ \underset{b}{I_2},\ \underset{-b}{3m+2},\ \underset{b}{I_3},\ \dots,\ \underset{b}{I_{m-1}},\ \underset{-b}{4m-1},\ \underset{b}{I_m}).$$

Obviously, $\sum_{\lambda=k}^{l} s_{\pi_\lambda}\leqslant b$ holds for all $1\leqslant k\leqslant l\leqslant n$.

Conversely, assume that P1 has a solution π with $\tilde{F}(\pi)\leqslant b$. In π the $m-1$ items $3m+1,\dots,4m-1$ can divide the remaining items $\{1,\dots,3m\}$ into at most m subsets I_j. If less than m subsets are separated by these items, at least one subset I_j exists with $|I_j|\geqslant 4$ and $F(\pi)\geqslant \sum_{i\in I_j} s_i > \sum_{i\in I_j} b/4 > 4\cdot b/4 = b$, which contradicts $\tilde{F}(\pi)\leqslant b$.

Thus, π must have the form $\pi=(I_1,3m+1,I_2,3m+2,I_3,\dots,I_{m-1},4m-1,I_m)$ with m subsets $I_1,\dots,I_m$ each containing three elements from $\{1,\dots,3m\}$. For all these sets I_j we have $\sum_{i\in I_j} s_i\leqslant b$, since $\tilde{F}(\pi)\leqslant b$. From $\sum_{j=1}^{m}\sum_{i\in I_j} s_i = mb$ we get the equality $\sum_{i\in I_j} s_i = b$. Thus, the sets $I_1,\dots,I_m$ define a solution of 3-PART. □

2.3. *Problem* $F2,R1|p_{ij}=p,t_j\in\{T_1,T_2\}|C_{\max}$

In this section we consider a special case of the problem discussed in the previous section where we have unit processing times on the machines and the job-dependent transportation times t_j may take only two rational values T_1 and T_2. W.l.o.g. assume $T_1<T_2$. If $T_1,T_2\leqslant 1$ or $T_1,T_2\geqslant 1$, each permutation is optimal, as observed in the previous subsection. Therefore, it remains to consider the case $T_1<1, T_2>1$.

For the considered problem the objective value of a permutation π can be calculated by

$$F(\pi)=\max_{k=1}^{n}\max_{l=k}^{n}\left\{\sum_{\lambda=k}^{l} s_{\pi_\lambda}+n+2\right\}=\max_{k=1}^{n}\max_{l=k}^{n}\{H^{\pi}(k,l)\} \tag{2.3}$$

with $H^{\pi}(k,l)=\sum_{\lambda=k}^{l} s_{\pi_\lambda}+n+2$ and $s_j:=t_j-1$ for $j=1,\dots,n$.

Let $S_1:=T_1-1<0$ and $S_2:=T_2-1>0$. A job with $s_j=S_2$ will be called an "S_2-job"; a job with $s_j=S_1$ will be called an "S_1-job". Since we have only two types of jobs, we may decompose a permutation π into "S_2"-blocks and "S_1-blocks", where a block consists of a maximal number of consecutive jobs in π with the same s_j-value. For example, the permutation π with $s_\pi=(S_2,S_2,S_1,S_2,S_2,S_2,S_1,S_1,S_2)$ contains the blocks $(S_2,S_2),(S_1),(S_2,S_2,S_2)$, (S_1,S_1), and (S_2).

If we have only a single S_2-job, obviously, each permutation π is optimal with objective value $F(\pi)=S_2+n+2$. Thus, we may restrict our considerations to the case in which we have at least two S_2-jobs. In the following lemmata we will show that the search for an optimal permutation π may be reduced to permutations with a special decomposition into blocks.

Lemma 3. *It is sufficient to consider only permutations π with $s_{\pi_1}=s_{\pi_n}=S_2$.*

Proof. Assume that $s_{\pi_1}=S_1$ and $s_\pi=(S_1,\ldots,S_1,S_2,\ldots)$. Let $\tilde{\pi}$ be the permutation we obtain by swapping the first S_1-block with the first S_2-job π_{k^*} leading to $s_{\tilde{\pi}}=(S_2,S_1,\ldots,S_1,\ldots)$. Then we have $H^{\tilde{\pi}}(1,1)=S_2+n+2=H^{\pi}(k^*,k^*)$ and $H^{\tilde{\pi}}(1,l)\leqslant H^{\tilde{\pi}}(1,1)=H^{\pi}(k^*,k^*)$ for all $l\leqslant k^*$. For all other values of k and l we get $H^{\tilde{\pi}}(k,l)\leqslant H^{\pi}(k,l)$ and therefore, $F(\tilde{\pi})=\max H^{\tilde{\pi}}(k,l)\leqslant\max H^{\pi}(k,l)=F(\pi)$. The case $s_{\pi_n}=S_1$ can be treated in the same way. □

Since $S_1<0$ and $S_2>0$, we have

Lemma 4. *The objective value* (2.3) *of a permutation π is achieved by a pair k,l, where π_k is the first job and π_l the last job of an S_2-block.*

Let q be the number of jobs j with $t_j=T_2$. Then we have $n-q$ jobs with $t_j=T_1$. In the following we distinguish two cases.

Case 1: $q>n/2$, i.e. we have more S_2-jobs than S_1-jobs.

Lemma 5. *Let $q>n/2$ and $b:=\lceil q/r\rceil$ with $r:=n-q+1$. Then an optimal permutation exists which contains r S_2-blocks of lengths b or $b-1$, separated by single S_1-jobs.*

Proof. Analogously to the proof of Lemma 3, we can show that we need not consider permutations in which two S_1-jobs are neighbored. Therefore we can conclude that we have $r=n-q+1$ S_2-blocks, separated by single S_1-jobs. Thus, π has the form $s_\pi=(B_1^\pi,S_1,B_2^\pi,S_1,\ldots,S_1,B_r^\pi)$ with S_2-blocks B_i^π of length b_i^π $(i=1,\ldots,r)$.

It remains to show that we only have blocks of lengths b and $b-1$. Assume that there is no optimal permutation with only blocks of these lengths. Then in each optimal permutation we have at least one S_2-block B_i^π with length $b_i^\pi\geqslant b+1$ or one block with $b_i^\pi<b-1$. Let π be an optimal permutation with a block of length $b_i^\pi\geqslant b+1$. Then there must be at least one block B_j^π of length $b_j^\pi\leqslant b-1$ in π. Choose the blocks B_i^π,B_j^π such that $|i-j|$ is minimal, i.e. all blocks between B_i^π and B_j^π (if any) have length b. Let us assume $j>i$; the other case can be treated in the same way.

Due to Lemma 4 the objective value of π can be written as

$$\begin{aligned} f(\pi) &= \max_{k=1}^{r}\ \max_{l=k}^{r}\left\{\sum_{\lambda=k}^{l}(b_\lambda^\pi\cdot S_2+S_1)-S_1+n+2\right\}\\ &=: \max_{k=1}^{r}\ \max_{l=k}^{r} A^\pi(k,l). \end{aligned} \tag{2.4}$$

Let $\tilde{\pi}$ be the permutation obtained from π by moving one S_2-job from B_i^π to B_j^π. Then

$$A^{\tilde{\pi}}(k,l) = \begin{cases} A^\pi(k,l), & k \leqslant i, l \geqslant j \text{ or } k > i, l < j \text{ or } k, l > j \text{ or } k, l < i, \\ A^\pi(k,l) - S_2, & k \leqslant i \leqslant l < j, \\ A^\pi(k,l) + S_2, & i < k \leqslant j \leqslant l. \end{cases}$$

Thus, we have $A^{\tilde{\pi}}(k,l) > A^\pi(k,l)$ only for $i < k \leqslant j \leqslant l$. However, for these cases we can find other terms which dominate the $A^{\tilde{\pi}}(k,l)$-values.

If $b \cdot S_2 + S_1 \geqslant 0$, then

$$A^{\tilde{\pi}}(k,l) = A^{\tilde{\pi}}(i,l) - \sum_{\lambda=i}^{k-1} (b_\lambda^{\tilde{\pi}} \cdot S_2 + S_1) \leqslant A^\pi(i,l) - (k-i)(b \cdot S_2 + S_1) \leqslant A^\pi(i,l).$$

If $b \cdot S_2 + S_1 < 0$, we distinguish the following three cases:

- $l > j$: $A^{\tilde{\pi}}(k,l) = A^\pi(j+1,l) + \sum_{\lambda=k}^{j} (b \cdot S_2 + S_1) \leqslant A^\pi(j+1,l)$.
- $l = j > k$: $A^{\tilde{\pi}}(k,j) = A^\pi(k,j-1) + (b \cdot S_2 + S_1) \leqslant A^\pi(k,j-1)$.
- $l = j = k$: $A^{\tilde{\pi}}(j,j) = b \cdot S_2 + n + 2 \leqslant (b+1) \cdot S_2 + n + 2 = A^\pi(i,i)$.

Therefore $F(\tilde{\pi}) = \max A^{\tilde{\pi}}(k,l) \leqslant \max A^\pi(k,l) = F(\pi)$.

The case in which a block B_i^π of length $b_i^\pi < b-1$ exists can be treated in a similar way. The result of the lemma follows by induction. □

Lemma 5 states that there is an optimal permutation in which the q S_2-jobs are partitioned into r blocks of lengths b and $b-1$. Thus, the numbers x_b and x_{b-1} of blocks with lengths b and $b-1$ are given by $x_{b-1}=rb-q$ and $x_b=n-rb+1$. Therefore the problem reduces to the problem of ordering r blocks, x_b with length b and x_{b-1} with length $b-1$, such that (2.4) is minimized. If we define $\hat{s}_j := b \cdot S_2 + S_1$ for all blocks of length b and $\hat{s}_j := (b-1) \cdot S_2 + S_1$ for all blocks of length $b-1$, we again have the problem of sequencing r numbers $\hat{s}_j \in \{\hat{S}_1, \hat{S}_2\}$ with $\hat{S}_1 := (b-1) \cdot S_2 + S_1 < \hat{S}_2 := b \cdot S_2 + S_1$ such that

$$F(\pi) = \max_{k=1}^{r} \max_{l=k}^{r} \left\{ \sum_{\lambda=k}^{l} \hat{s}_{\pi_\lambda} + n + 2 \right\} - S_1, \tag{2.5}$$

is minimized. The objective functions (2.3) and (2.5) differ only in the constant S_1. Therefore the problem can be solved recursively.

Case 2: $q \leqslant \frac{n}{2}$

Lemma 6. Let $q \leqslant n/2$ and $b' := \lceil (n-q)/r' \rceil$ with $r' := q-1$. Then an optimal permutation exists which contains r' S_1-blocks of lengths b' or $b'-1$, separated by single S_2-jobs.

Proof. Analogously to Case 1 we can show that we need not consider permutations in which two S_2-jobs are neighbored. Therefore, we can conclude that we have $r' = q-1$ S_1-blocks, separated by single S_2-jobs. Thus, π has the form $s_\pi = (S_2, A_1^\pi, S_2, A_2^\pi, S_2, \ldots, S_2, A_{r'}^\pi, S_2)$ with S_1-blocks A_i^π of length a_i^π.

The objective value of π can be written as

$$F(\pi) = \max_{k=1}^{r'} \max_{l=k}^{r'} \left\{ \sum_{\lambda=k}^{l} (a_\lambda^\pi \cdot S_1 + S_2) + S_2 + n + 2 \right\}. \tag{2.6}$$

The proof that it is sufficient to consider only permutations with blocks of lengths b' and $b'-1$ can be done analogously to the proof of Lemma 5. □

Lemma 6 states that there is an optimal permutation in which the $n-q$ S_1-jobs are partitioned into r' blocks of lengths b' and $b'-1$. Thus, the numbers $y_{b'}$ and $y_{b'-1}$ of blocks with lengths b' and $b'-1$ are given by $y_{b'-1}=r'b'+q-n$ and $y_{b'}=n-r'b'-1$. Therefore the problem reduces to the problem of ordering r' blocks, $y_{b'}$ with length b' and $y_{b'-1}$ with length $b'-1$, such that (2.6) is minimized. If we define $\hat{s}_j:=b'\cdot S_1+S_2$ for all blocks of length b' and $\hat{s}_j:=(b'-1)\cdot S_1+S_2$ for all blocks of length $b'-1$, we have to sequence r' numbers $\hat{s}_j\in\{\hat{S}_1,\hat{S}_2\}$ with $\hat{S}_1:=b'\cdot S_1+S_2<\hat{S}_2:=(b'-1)\cdot S_1+S_2$ such that

$$F(\pi)=\max_{k=1}^{r'}\max_{l=k}^{r'}\left\{\sum_{\lambda=k}^{l}\hat{s}_{\pi_\lambda}+n+2\right\}+S_2 \tag{2.7}$$

is minimized. The objective functions (2.3) and (2.7) differ only in the constant S_2. Therefore, also in the second case the problem can be solved recursively.

Using these results we formulate the following recursive procedure `Blocks` $(n,q,S_1,S_2,\sigma_1,\sigma_2)$, which solves our problem. If a permutation π has the form $\pi=(\hat{\pi},\sigma_1)$, then let `Deletelast` $(\pi,\sigma_1)=\hat{\pi}$.

```
PROCEDURE Blocks (n,q,S1,S2,σ1,σ2)
 1.     IF n = 1 OR q = 1 OR S1,S2 ≥ 0 OR S1,S2 ≤ 0 THEN
            RETURN an arbitrary permutation of the q sequences
            σ2 and the n − q sequences σ1;
 2.     IF q > n/2 THEN BEGIN
 3.         r:=n − q + 1; b:=⌈q/r⌉;
 4.         xb:=n − rb + 1; xb−1:=rb − q;
 5.         Ŝ1:=(b − 1)·S2 + S1; Ŝ2:=b·S2 + S1;
 6.         σ̂1:=(σ2,...,σ2 , σ1); σ̂2:=(σ2,...,σ2 , σ1);
                 (b−1)            (b)
 7.         π:=Blocks (r,xb,Ŝ1,Ŝ2,σ̂1,σ̂2);
 8.         π:=Deletelast (π,σ1);
        END
 9.     ELSE BEGIN
10.         r':=q − 1; b':=⌈(n−q)/r'⌉;
11.         yb':=n − r'b' − 1; yb'−1:=r'b' + q − n;
12.         Ŝ1:=b'·S1 + S2; Ŝ2:=(b' − 1)·S1 + S2;
13.         σ̂1:=(σ1,...,σ1 , σ2); σ̂2:=(σ1,...,σ1 , σ2);
                 (b')            (b'−1)
14.         π:=Blocks (r', yb'−1, Ŝ1, Ŝ2, σ̂1, σ̂2);
15.         π:=(σ2,π);
        END
16.     RETURN π
```

Theorem 7. *Procedure* `Blocks` $(n, q, T_1-1, T_2-1, (S_1), (S_2))$ *solves problem* $F2, R1|p_{ij}=1, t_j \in \{T_1, T_2\}|C_{\max}$ *in* $O(\log n)$ *time.*

Proof. The optimality follows from the preceding considerations. It remains to prove the complexity. In Case 1 we have $r = n - q + 1 < n - n/2 + 1 = O(n/2)$. In Case 2 we have $r' = q - 1 < n/2 - 1 = O(n/2)$. Thus, in each case the number of jobs reduces by the factor 2. Therefore procedure `Blocks` $(n, q, S_1, S_2, \sigma_1, \sigma_2)$ which can be executed in constant time is called at most $O(\log n)$ times. Since the input length of the problem is $O(\log n)$, the algorithm is polynomial. □

Example 8. Let $n = 78$, $q = 33$, $T_1 = 0$ and $T_2 = 2.4$.

1. Iteration: `Blocks` $(78, 33, -1.0, 1.4, (S_1), (S_2))$: $q \leqslant n/2$, i.e. we get Case 2:

   $$r' = 32, b' = 2, y_{b'} = 13, y_{b'-1} = 19, \hat{S}_1 = -0.6, \hat{S}_2 = 0.4.$$

   This yields 13 S_1-blocks of length 2 and 19 of length 1. These 32 blocks are separated by the 33 S_2-jobs. Let $\sigma_1^1 = (S_1, S_1, S_2)$ and $\sigma_2^1 = (S_1, S_2)$. Then the optimal permutation in this step has the form $\pi^1 = (S_2, \pi^2)$ where π^2 is the optimal permutation of sequences σ_1^1, σ_2^1 calculated in the next iteration.
2. Iteration: `Blocks` $(32, 19, -0.6, 0.4, \sigma_1^1, \sigma_2^1)$: $q > n/2$, i.e. we get Case 1:

   $$r = 14, b = 2, x_b = 5, x_{b-1} = 9, \hat{S}_1 = -0.2, \hat{S}_2 = 0.2.$$

   This yields 5 σ_2^1-blocks of length 2 and 9 of length 1. These 14 blocks are separated by the 13 σ_1^1-blocks. Let $\sigma_1^2 = (\sigma_2^1, \sigma_1^1)$ and $\sigma_2^2 = (\sigma_2^1, \sigma_2^1, \sigma_1^1)$. Then the optimal permutation in this step has the form $\pi^2 =$ `Deletelast` (π^3, σ_1^1) where π^3 is the optimal permutation of sequences σ_1^2, σ_2^2 calculated in the next iteration.
3. Iteration: `Blocks` $(14, 5, -0.2, 0.2, \sigma_1^2, \sigma_2^2)$: $q \leqslant n/2$, i.e. we get Case 2:

   $$r' = 4, b' = 3, y_{b'} = 1, y_{b'-1} = 3, \hat{S}_1 = -0.4, \hat{S}_2 = -0.2.$$

   This yields 1 σ_1^2-block of length 3 and 3 of length 2. These 4 blocks are separated by the 5 σ_2^2-blocks. Let $\sigma_1^3 = (\sigma_1^2, \sigma_1^2, \sigma_1^2, \sigma_2^2)$ and $\sigma_2^3 = (\sigma_1^2, \sigma_1^2, \sigma_2^2)$. Then the optimal permutation in this step has the form $\pi^3 = (\sigma_2^2, \pi^4)$ where π^4 is the optimal permutation of sequences σ_1^3, σ_2^3 calculated in the next iteration.
4. Iteration: `Blocks` $(4, 3, -0.4, -0.2, \sigma_1^3, \sigma_2^3)$: Since $S_1, S_2 \leqslant 0$, the recursion terminates and each permutation of the current sequences is optimal. We may set $\pi^4 = (\sigma_1^3, \sigma_2^3, \sigma_2^3, \sigma_2^3)$ and obtain the optimal permutation $\pi = \pi^1$.

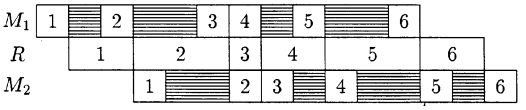
2.4. *Problem* $F2, R1|p_{ij} = 1, t_j|C_{\max}$

In this section we consider another special case of problem $F2, R1|p_{ij} = p, t_j|C_{\max}$ where $p = 1$. All transportation times t_j are supposed to be non-negative integers. If $t_j > 0$ for all jobs j holds, we have $\min_{j=1}^{n} \{t_j\} \geqslant 1$ due to integrality. Thus, in this case the transportation times dominate the processing times as discussed before and each permutation is optimal. In the other case where some t_j-values are zero, we assume

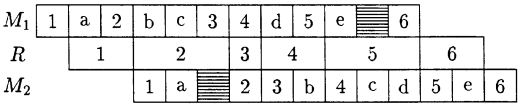
that $t_j = 0$ does not mean that job j does not have to be transported, but means that the job needs the robot for a very small time, i.e. the robot has to be available for transportation between processing on M_1 and M_2.

To solve the problem we divide the jobs j into two disjoint sets $J^+ := \{j | t_j > 0\}$ and $J^0 := \{j | t_j = 0\}$. In the following we describe an algorithm which constructs an optimal permutation plan. The algorithm is illustrated by an example where the jobs from J^+ are numbered from $1, \ldots, 6$ and the jobs from J^0 by a, ..., e.

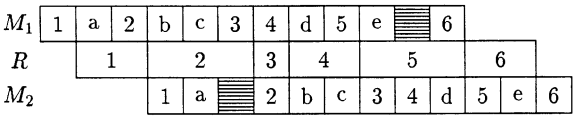
(1) Choose an arbitrary permutation π^+ of the jobs from J^+. Plan all jobs in such a way that the robot transports without idle time and such that each job is processed on M_1 immediately before transportation and on M_2 immediately after transportation.



(2) Insert all jobs from J^0 into the empty time periods in π^+ on M_1 and as early as possible into the empty time periods on M_2. If there are more jobs than empty time periods on M_1, put the remaining jobs to the end of π^+.



(3) Reorder the jobs on M_2 according to the permutation π of the jobs on M_1, resulting in a permutation plan with the same makespan.



The makespan of the schedule resulting after Step 1 provides a lower bound for the optimal makespan. It is given by $F(\pi^+) = 2 + \sum_{j \in J^+} t_j$.

Theorem 9. *The described algorithm constructs an optimal permutation π for problem $F2, R1 | p_{ij} = 1, t_j | C_{\max}$ with makespan*

$$F(\pi) = \max \left\{ 2 + \sum_{j \in J^+} t_j, \max_{j \in J^+} t_j + n + 1 \right\}.$$

Proof. Consider the schedule after Step 2 (which has the same makespan as the final schedule). Let l be the last job from J^+ which has an idle time immediately before its processing on M_2.

Case 1: No job of J^0 starts after l on M_2. In this case the makespan of the schedule is given by the makespan of the schedule after Step 1, i.e.

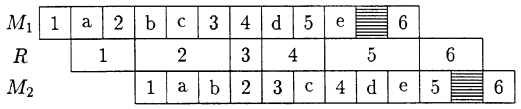
$$F(\pi) = 2 + \sum_{j \in J^+} t_j. \tag{2.8}$$

Case 2: Jobs of J^0 are scheduled after l on M_2. By construction all jobs scheduled before l on M_1 are also scheduled before l on M_2 (due to the idle time before l). Thus, jobs of J^0 are also scheduled after l on M_1 and no idle times occur before l on M_1. Therefore, the makespan of the schedule after Step 2 is given by the complete processing and transportation of job l plus the number of jobs scheduled before l on M_1 plus the number of jobs scheduled after l on M_2. Since each job $j \neq l$ is either scheduled before l on M_1 or after l on M_2 (but not both), we get

$$F(\pi) = 1 + t_l + 1 + (n-1) = t_l + n + 1. \tag{2.9}$$

Furthermore, $F(\pi) \geqslant t_j + n + 1$ for all $j \in J^+$, i.e. $F(\pi) = \max_{j \in J^+} t_j + n + 1$. Since (2.8) and (2.9) are lower bounds for the optimal makespan, the constructed schedule is optimal. □

Remark. If $t_j = 0$ means that the robot is not needed for the transportation of job j, an optimal permutation plan does not necessarily exist. Obviously, Steps (1) and (2) of our algorithm solve the problem. Note that by planning the jobs with $t_j = 0$ as early as possible in the empty time periods on M_2 in Step (2), some jobs may overtake others during their transportation. For the example from above we get the following schedule where jobs b and d overtake jobs 2 and 5, respectively:



3. Flow-shop problems with *m* machines

In this section we consider a problem in which we have an arbitrary number m of machines. Since problem $F, R1 \mid p_{ij} = p, t_{jk} \mid C_{\max}$ is already $\mathcal{NP}$-hard for $m = 2$, it is only of interest to study problems with job-independent transportation times $t_{jk} = t_k$. We will show that problem $F, R1 \mid n \geqslant m - 1, p_{ij} = p, t_{jk} = t_k \mid C_{\max}$ or equivalently $F, R1 \mid n \geqslant m-1, p_{ij} = 1, t_{jk} = t_k$ rational $\mid C_{\max}$ is polynomially solvable. In this problem we assume $n \geqslant m - 1$, i.e. the number of jobs is greater than or equal to the number of transportation stages. From the practical point of view, this case is also the most relevant case, since in general the number of jobs will exceed the number of machines.

It is sufficient to consider only permutation plans, since the jobs are identical and overtaking does not reduce the completion times. Furthermore, the objective value is independent of the permutation and depends only on the sequence of the robot. We

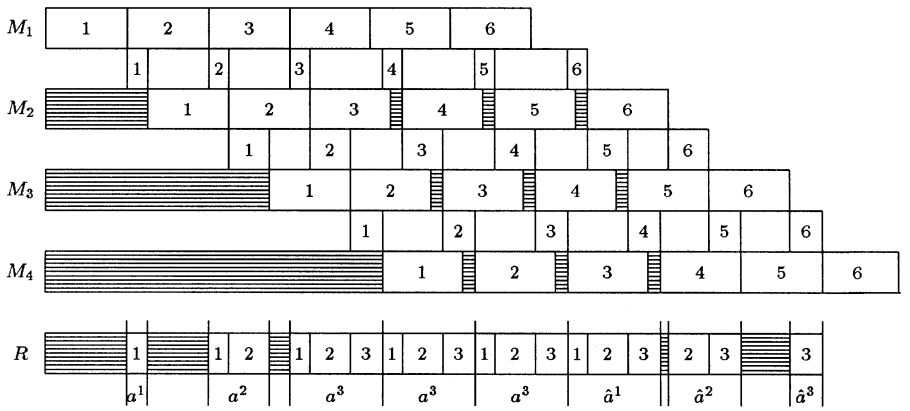


Fig. 1. The robot sequence Z^*.

describe the robot sequence by a vector $Z = (z_1, \ldots, z_{n(m-1)})$ with $z_\lambda = k$ iff the λth move of the robot transports the next available job from M_k to M_{k+1}. The corresponding schedule is obtained by starting all jobs on the machines as early as possible.

For $k = 1, \ldots, m-1$ let $a^k := (1, \ldots, k)$ and $\hat{a}^k := (k, \ldots, m-1)$ be the special sequences in which the robot serves the machines in the order $1, 2, \ldots, k$ and $k, k+1, \ldots, m-1$, respectively. In the following we will show that there exists an optimal robot sequence which contains only these subsequences. Let

$$Z^* := (a^1, a^2, \ldots, a^{m-2}, \underbrace{a^{m-1}, a^{m-1}, \ldots, a^{m-1} = \hat{a}^1}_{n-m+2}, \hat{a}^2, \hat{a}^3, \ldots, \hat{a}^{m-1}).$$

Z^* consists of

- a "building-up" phase where in each step one more machine is served,
- an "identity" phase where all machines are served regularly, and
- a "building-down" phase where in each step one machine less is served.

In Fig. 1 an example for this robot sequence with $n = 6$ and $m = 4$ is given, which in general is not a first-come-first-served sequence. The next theorem proves that this sequence is optimal.

Theorem 10. *The sequence Z^* defines an optimal robot sequence for problem $F, R1 \mid n \geqslant m-1, p_{ij} = p, t_{jk} = t_k \mid C_{\max}$.*

Proof. Let $l \in \{0, \ldots, m-1\}$ be the index with $\sum_{\lambda=1}^{l} t_\lambda \leqslant 1$ and $\sum_{\lambda=1}^{l+1} t_\lambda > 1$, where $l = 0$ if $t_1 > 1$ and $l = m-1$ if $\sum_{k=1}^{m-1} t_k \leqslant 1$. In the following we will calculate the objective value of the schedule associated with Z^*.

At first we consider the sequence $(a^1, a^2, \ldots, a^l)$. The transportations of a^1 can be done in the interval $[1, 1+t_1]$, those of a^2 in $[2, 2+t_1+t_2]$, and those of a^k in $[k, k+\sum_{\lambda=1}^{k} t_\lambda]$ for $k = 1, \ldots, l$. When the robot has transported a job from M_j to M_{j+1} it is back at M_j after at most one time unit. Thus, the next job completed on M_j after one time unit does not have to wait for the robot, i.e. we have a no-wait plan. Only the

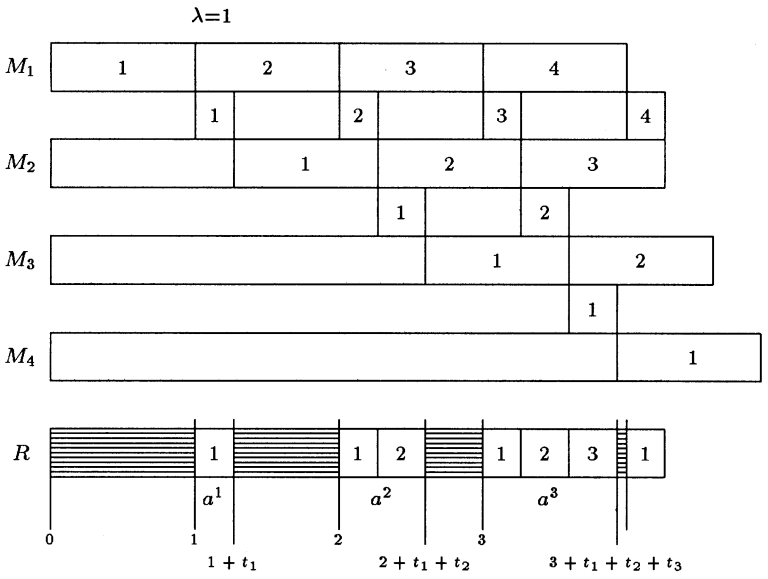


Fig. 2. The no-wait part of robot sequence Z^*.

robot has a waiting time between a^k and a^{k+1} of length $\tau_k := 1 - \sum_{\lambda=1}^{k} t_\lambda$ for $k = 1, \ldots, l$ (cf. Fig. 2).

Case 1: $l = m - 1$

In this case each job in the partial sequence

$$\underbrace{(a^{m-1}, \ldots, a^{m-1})}_{n-m+2}$$

is again transported immediately after finishing on a machine, i.e. we have a no-wait plan.

Case 2: $l < m - 1$

We consider the sequence $(a^{l+1}, \ldots, a^{m-1})$. The transportations of a^{l+1} can be done in the interval $[l+1, l+1+\sum_{\lambda=1}^{l+1} t_\lambda]$, those of a^{l+2} in

$$\left[l+1+\sum_{\lambda=1}^{l+1} t_\lambda, l+1+\sum_{\lambda=1}^{l+1} t_\lambda + \sum_{\lambda=1}^{l+2} t_\lambda \right],$$

and those of a^{l+k} in

$$\left[l+1+\sum_{\mu=1}^{k-1}\sum_{\lambda=1}^{l+\mu} t_\lambda, l+1+\sum_{\mu=1}^{k}\sum_{\lambda=1}^{l+\mu} t_\lambda \right] \quad \text{for } k = 1, \ldots, m-1-l,$$

since $\sum_{\lambda=1}^{l+1} t_\lambda > 1$. This means that the robot transports jobs in the sequence

$$(a^{l+1},\ldots,a^{m-2},\underbrace{a^{m-1},a^{m-1},\ldots,a^{m-1}=\hat{a}^1}_{n-m+2})$$

without idle time and all machines are served regularly.

Symmetrically, let $\hat{l}\in\{0,\ldots,m-1\}$ be the index with $\sum_{\lambda=m-\hat{l}}^{m-1} t_\lambda \leqslant 1$ and $\sum_{\lambda=m-\hat{l}-1}^{m-1} t_\lambda > 1$, where $\hat{l}=0$ if $t_{m-1}>1$.

As above we can show that the robot is always busy in the sequence $(\hat{a}^1,\ldots,\hat{a}^{m-\hat{l}-1})$. Furthermore, from index $m-\hat{l}$ up to index $m-1$ we have a no-wait plan and the robot only has waiting times $\hat{\tau}_k := 1-\sum_{\lambda=k}^{m-1} t_\lambda$ for $k=1,\ldots,\hat{l}$ between $\hat{a}^{m-k-1}$ and $\hat{a}^{m-k}$.

In the first case ($l=m-1$), the whole schedule is a no-wait schedule in which each job is transported immediately after processing, i.e.

$$F(Z^*)=n+m-1+\sum_{k=1}^{m-1} t_k.$$

Since a no-wait schedule provides a lower bound for the optimal makespan, Z^* must be optimal.

In the second case ($l<m-1$), Z^* results in the following working and waiting time sequence for the robot:

$$(a^1,\tau_1,a^2,\ldots,a^l,\tau_l,a^{l+1},a^{l+2},\ldots,a^{m-2},\underbrace{a^{m-1},\ldots,a^{m-1}=\hat{a}^1}_{n-m+2},$$
$$\hat{a}^2,\ldots,\hat{a}^{m-\hat{l}-1},\hat{\tau}_l,\hat{a}^{m-\hat{l}},\ldots,\hat{a}^{m-2},\hat{\tau}_1,\hat{a}^{m-1}).$$

Its objective value consists of processing the first job on M_1, the last job on M_m, all transportation times and some waiting times for the robot, i.e.

$$F(Z^*)=2+n\sum_{k=1}^{m-1} t_k+\sum_{k=1}^{l}\tau_k+\sum_{k=1}^{\hat{l}}\hat{\tau}_k.$$

We claim that all waiting times $\sum_{k=1}^{l}\tau_k$ and $\sum_{k=1}^{\hat{l}}\hat{\tau}_k$ in $F(Z^*)$ are unavoidable. We have shown that the first sum $\sum_{k=1}^{l}\tau_k$ results from a no-wait schedule for all transported jobs in the sequence $(a^1,\ldots,a^l)$. Even if we had an unlimited number of robots, we could not improve this first part of the schedule since each job is already transported as early as possible, i.e. in each feasible schedule the robot has to wait $\sum_{k=1}^{l}\tau_k$ time units in the first part of the schedule. Thus, the waiting times for the single robot are unavoidable. Symmetrically, the waiting times $\sum_{k=1}^{\hat{l}}\hat{\tau}_k$ are unavoidable, i.e. Z^* is optimal. □

Remark. For the case $n<m-1$ the proof of the result in Theorem 10 cannot be adapted. It is an open question whether an analogous result also holds for this case (we did not succeed in proving such a result, but neither did we find a counterexample).

Finke et al. [3] considered a robotic flow-shop cell with a single unit of buffer behind each machine where all jobs are identical, i.e. $p_{ij}=p_i$ holds. They showed that repeating the sequence a^{m-1} from the middle part of Z^* minimizes the mean cycle time

Table 1
Complexity results for flow-shop problems with a single robot

Polynomially solvable:	
$F2,R1\|p_{ij}=1,t_j\|C_{\max}$ $F3\|p_{1j}=p_{3j}=1\|C_{\max}$	Section 2.4
$F2,R1\|p_{ij}=p,t_j\in\{T_1,T_2\}\|C_{\max}$	Section 2.3
$F,R1\|n\geqslant m-1,\ p_{ij}=p,t_{jk}=t_k\|C_{\max}$	Section 3
strongly $\mathcal{NP}$-hard:	
$F2,R1\|p_{ij}=p,t_j\|C_{\max}$ $F3\|p_{1j}=p_{3j}=p\|C_{\max}$	Section 2.2
$F2,R1\|t_j=T\|C_{\max}$ $F3\|p_{2j}=p_2\|C_{\max}$	Kise [8], Section 2.1
$F2\|r_j\|C_{\max}$	Lenstra et al. [10]
$F3\|\|C_{\max}$	Garey et al. [4]

C_t in repetitive manufacturing where initially all machines are occupied by a job. Note that in our algorithm a single unit of buffer behind each machine is also sufficient. The main difference between the two results is the objective function. While minimizing the cycle time C_t in an environment with $p_{ij}=p$ becomes trivial ($C_t=\max\{p,\sum_{k=1}^{m-1}t_k\}$), it is not obvious how to sequence the robot in a schedule with minimal makespan when not all machines are initially occupied by a job. Note that the sequence Z^* is not makespan-optimal in the more general case $p_{ij}=p_i$.

4. Concluding remarks

We have derived new complexity results for flow-shop problems with transportation times and a single robot and the objective to minimize the makespan. Some of these may also be interpreted as new results for special cases of the classical 3-machine flow-shop $F3\|\|C_{\max}$ with constant processing times at certain stages. These results are summarized in Table 1.

It may be interesting to study some generalizations of the considered problems, for example, problems with

- different release dates for the jobs,
- two, three or any limited number of robots,
- empty moving times for the robot, or
- other objective functions.

All known complexity results as well as open problems in the class of flow-shop problems with transportation times and a single robot can be found at the web-site
`http://www.mathematik.uni-osnabrueck.de/research/OR/class/`.

Acknowledgements

We gratefully acknowledge the constructive comments of the anonymous referees.

References

[1] Y. Crama, J. van de Klundert, Cyclic scheduling of identical parts in a robotic cell, Oper. Res. 45 (1997) 952–965.
[2] Y. Crama, V. Kats, J. van de Klundert, E. Levner, Cyclic scheduling in robotic flowshops, Working paper (1997), Ann. Oper. Res., to appear.
[3] G. Finke, C. Gueguen, N. Brauner, Robotic cells with buffer space, Conference Proceedings of ECCO IX, Dublin, 1996.
[4] M. Garey, D.S. Johnson, R. Sethi, The complexity of flow-shop and job-shop scheduling, Math. Oper. Res. 1 (1976) 117–129.
[5] R. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy-Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Ann. Discrete Math. 5 (1979) 287–326.
[6] N. Hall, H. Kamoun, C. Sriskandarajah, Scheduling in robotic cells: Classification, two and three machine cells, Oper. Res. 45 (1997) 421–439.
[7] S.M. Johnson, Optimal two- and three-stage production schedules with setup times included, Naval Res. Logist. Quart. 1 (1954) 61–68.
[8] H. Kise, On an automated two-machine flowshop scheduling problem with infinite buffer, J. Oper. Res. Soc. Japan 34 (1991) 354–361.
[9] H. Kise, T. Shioyama, T. Ibaraki, Automated two-machine flowshop scheduling: a solvable case, IIE Trans. 23 (1991) 10–16.
[10] J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems, Ann. Discrete Math. 1 (1977) 343–362.
[11] E. Levner, K. Kogan, I. Levin, Scheduling a two-machine robotic cell: a solvable case, Ann. Oper. Res. 57 (1995) 217–232.
[12] L.G. Mitten, Sequencing n jobs on two machines with arbitrary time lags, Mg. Sci. 5 (1958) 293–298.
[13] S.S. Panwalkar, Scheduling of a two-machine flowshop with travel time between machines, J. Opl. Res. Soc. 42 (1991) 609–613.
[14] S. Sethi, C. Sriskandarajah, G. Sorger, J. Blazewicz, W. Kubiak, Sequencing of parts and robot moves in a robotic cell, The Internat. J. Flexible Manuf. Systems 4 (1992) 331–358.
[15] H. Stern, G. Vitner, Scheduling parts in a combined production-transportation work cell, J. Opl. Res. Soc. 41 (1990) 625–632.
[16] W. Yu, The two-machine flow shop problem with delays and the one-machine total tardiness problem, Ph.D. Thesis, Technische Universiteit Eindhoven, 1996.