ZOR – Mathematical Methods of Operations Research (1994) 39:305–319

On Non-Permutation Solutions to Some Two Machine Flow Shop Scheduling Problems

VITALY A. STRUSEVICH

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

CARIN M. ZWANEVELD

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

Abstract: In this paper, we study two versions of the two machine flow shop scheduling problem, where schedule length is to be minimized. First, we consider the two machine flow shop with setup, processing, and removal times separated. It is shown that an optimal solution need not be a permutation schedule, and that the problem is NP-hard in the strong sense, which contradicts some known results. The tight worst-case bound for an optimal permutation solution in proportion to a global optimal solution is shown to be 3/2. An $O(n)$ approximation algorithm with this bound is presented. Secondly, we consider the two machine flow shop with finite storage capacity. Again, it is shown that there may not exist an optimal solution that is a permutation schedule, and that the problem is NP-hard in the strong sense.

Key Words: flow shop, permutation schedules, complexity, approximation

1 Introduction

In this paper, we consider two versions of the two machine flow shop scheduling problem to minimize the schedule length. First, we consider the two machine flow shop with setup, processing, and removal times separated. Secondly, we consider the two machine flow shop with finite storage capacity. In both cases it is shown that, in contrary to what was believed before, there may not exist an optimal solution among *permutation* schedules, i.e., schedules with the same ordering of jobs on all machines.

The *two machine flow shop problem with setup, processing, and removal times separated* can be described as follows. Each job J_j, $j = 1, 2, \ldots, n$, consists of a chain of operations $(O_{1,j}, O_{2,j})$, which are to be performed in that order on the machines M_1 and M_2, respectively. Each operation $O_{i,j}$, $i = 1, 2; j = 1, 2, \ldots, n$, consists of three stages: a setup, a processing, and a removal stage. The setup stage precedes the processing stage, and the removal stage follows the processing stage. The setup of an operation can only start after the removal stage of its predecessor on that machine has been completed. Once the setup has started, the processing and the removal stages of that operation must follow without being interrupted by other operations. The setup, the processing, and the removal stages of an operation $O_{i,j}$, $i = 1, 2$, take $s_{i,j}$, $p_{i,j}$, and $r_{i,j}$ time units, respectively. A machine can perform only one operation, and one stage, at a time. Any stage of $O_{2,j}$ can be performed during the removal stage of $O_{1,j}$. Similarly, the setup stage of $O_{2,j}$ can be performed during any stage of $O_{1,j}$. The flow shop assumption can be formulated here as that the processing stage of $O_{2,j}$ must start not before the processing stage of $O_{1,j}$ has been completed, for all J_j.

We now turn to the description of the *two machine flow shop problem with finite storage capacity.* The formulation for the classical flow shop problem is applied here. An operation consists of processing only, a machine can perform only one operation at a time, and the flow shop assumption here implies that $O_{2,j}$ must start not before $O_{1,j}$ has been completed, for all J_j.

Furthermore, some extra restrictions arise from the finite storage capacity. During the processing of $O_{1,j}$ on M_1, a half-product is generated. Further processing of this half-product has to be done on M_2; this defines $O_{2,j}$. After completion of $O_{1,j}$, the half-product must be stored in the buffer until $O_{2,j}$ may start on M_2. If the buffer is full, then the half-product must wait on M_1, and this prevents this machine from performing the next operation. In the classical flow shop, there is no restriction on the storage capacity between the two machines, i.e., it is assumed that the buffer capacity is sufficiently large. We define the buffer capacity b as the number of half-products generated on M_1 that the buffer can contain. If $1 \leq b \leq n - 2$ then the buffer capacity may be a restriction. This case is referred to as finite buffer capacity.

For both versions of the two machine flow shop we assume the following. All J_j have zero release times, i.e., they may start at time zero. Without loss of generality, we assume that all values are integral. No preemption is allowed, i.e., once started, a stage of an operation cannot be interrupted before completion.

The criterion for optimality is the makespan C_{max}, i.e., it is required to minimize the time that both machines have completed all n jobs.

We adopt the notation $F2| \, |C_{max}$ for the classical two machine flow shop, as used by Lawler, Lenstra, Rinnooy Kan and Shmoys (1993). The version of that problem with setup, processing, and removal times separated is denoted by $F2|s_{ij}, r_{ij}|C_{max}$. The flow shop problem with finite buffer capacity is denoted by $F2|b|C_{max}$, and the buffer capacity may be specified, for example, to be equal to 1 by $F2|b = 1|C_{max}$.

The problem $F2|s_{ij}, r_{ij}|C_{max}$ was considered by Sule (1982) and Sule and Huang (1983), who claimed that an optimal solution could be found in polynomial time by using the Johnson algorithm for an artificial $F2| \, |C_{max}$ problem with specially defined processing times of the operations $O_{1,j}$ and $O_{2,j}$. They followed the way of reasoning of Yoshida and Hitomi (1979) who derived a similar result for the flow shop with setup times separated only. Sule and Huang came to their conclusion, because they took for granted that, for

$F2|s_{ij}, r_{ij}|C_{max}$, there would always exist an optimal solution that was a permutation schedule, which is not true in general, as we show below. We consider the problem $F2|s_{ij}, r_{ij}|C_{max}$ in Section 2.

In Section 3, we consider the problem $F2|b|C_{max}$, $1 \leq b \leq n-2$. This problem was studied by Papadimitriou and Kanellakis (1980), although they restricted themselves to a *first-in, first-out* (*FIFO*) buffer policy, i.e., they assumed that a job does not leave the buffer before its predecessors left it. Thus, only permutation schedules were considered. They proved this problem to be NP-hard in the strong sense, and presented an approximation algorithm with a tight worst-case ratio bound of $(2b+1)/(b+1)$. We show that under arbitrary buffer policy, $F2|b|C_{max}$ may have only non-permutation optimal solutions, and the problem remains NP-hard in the strong sense. The Papadimitriou-Kanellakis approximation algorithm is still applicable to the latter problem, and the worst-case bound does not change.

2 The Problem $F2|s_{ij}, r_{ij}|C_{max}$

For the classical flow shop with two or three machines, $F2|\ |C_{max}$ and $F3|\ |C_{max}$, there always exists an optimal permutation schedule (see Conway, Maxwell and Miller (1967)). For the flow shop with setup times separated, the same property holds for two machines only as proved by Yoshida and Hitomi (1979).

The problem $F2|\ |C_{max}$ is solvable in $O(n \log n)$ time due to Johnson (1954), while $F3|\ |C_{max}$ is NP-hard in the strong sense (see Garey, Johnson and Sethi (1976)). The problem $F2|s_{ij}|C_{max}$ is solvable in $O(n \log n)$ time by using the Johnson algorithm for the classical two machine flow shop problem, where the processing times of the operations $O_{1,j}$ and $O_{2,j}$ of job J_j are equal to $s_{1,j} + p_{1,j} - s_{2,j}$ and $p_{2,j}$, respectively (see Yoshida and Hitomi (1979)).

As mentioned before, $F2|s_{ij}, r_{ij}|C_{max}$ was considered by Sule (1982) and by Sule and Huang (1983), who claimed that the optimal solution could be found in $O(n \log n)$ time by using the Johnson algorithm for the classical two machine flow shop problem where the processing times of the operations O_{1j} and O_{2j} are equal to $s_{1,j} + p_{1,j} - s_{2,j}$ and $p_{2,j} + r_{2,j} - r_{1,j}$, respectively. However, as proved, this result is only correct for $F2|s_{ij}, r_{ij}|C_{max}$ restricted to *permutation* solutions.

Define

$$G_{i,j} = s_{i,j} + p_{i,j} + r_{i,j}\ , \qquad i = 1, 2\ .$$

For a schedule S, let $C_{1,j}(S)$, $j = 1, 2, \ldots, n$, denote the completion time of operation $O_{1,j}$; $C_{2,j}(S)$ is defined analogously.

Let S be a schedule associated with a permutation π. Without loss of generality, assume that $\pi = (1, 2, \ldots, n)$. As shown by Sule (1982), Sule and Huang (1983),

$$C_{2,n}(S) = \max\left\{\max_{1 \le u \le n}\left\{\sum_{j=1}^{u}(s_{1,j} + p_{1,j} - s_{2,j}) - \sum_{j=1}^{u-1}(p_{2,j} + r_{2,j} - r_{1,j})\right\}, 0\right\}$$
$$+ \sum_{j=1}^{n} G_{2,j} \,.$$

For each job $J_j, j = 1, 2, \ldots, n$, denote

$$a_j = s_{1,j} + p_{1,j} - s_{2,j}\,; \qquad b_j = p_{2,j} + r_{2,j} - r_{1,j}\,. \tag{2.1}$$

Since $C_{max}(S) = \max\{C_{1,n}(S), C_{2,n}(S)\}$, and $C_{1,n}(S) = \sum_{j=1}^{n} G_{1,j}$, the following statement holds.

Lemma 2.1: For $F_2|s_{ij}, r_{ij}|C_{max}$ restricted to permutation solutions, the makespan of a schedule S associated with a permutation of jobs $\pi = \{j, j_2, \ldots, j_n\}$ is specified by

$$C_{2,n}(S) = \max\left\{\sum_{k=1}^{n} G_{1,j_k}, \sum_{k=1}^{n} G_{2,j_k}, \max_{1 \le u \le n}\left\{\sum_{k=1}^{u} a_{j_k} - \sum_{k=1}^{u-1} b_{j_k}\right\} + \sum_{k=1}^{n} G_{2,j_k}\right\}. \tag{2.2}$$

It follows directly from Lemma 2.1, due to similarities in the expressions for C_{max}, that the Johnson algorithm for $F2|\ |C_{max}$ can be used to find the best permutation solution for $F2|s_{ij}, r_{ij}|\,C_{max}$.

We now show that, for $F2|s_{ij}, r_{ij}|\,C_{max}$, the search for an optimal solution should not be restricted to considering only permutation schedules.

Lemma 2.2: For $F2|s_{ij}, r_{ij}|\,C_{max}$, there may not be an optimal solution that is a permutation schedule.

Proof: We present an example where the unique optimal solution is not a permutation schedule. There are two jobs J_1 and J_2 such that

$$s_{1,1} = 0,\quad p_{1,1} = 1,\quad r_{1,1} = 0,\quad s_{2,1} = 0,\quad p_{2,1} = 1,\quad r_{2,1} = 0\ ;$$

$$s_{1,2} = 0,\quad p_{1,2} = 1,\quad r_{1,2} = 4,\quad s_{2,2} = 3,\quad p_{2,2} = 2,\quad r_{2,2} = 0\ .$$

Consider the schedule shown in Fig. 2.1. It is easy to check that this schedule with the job order (J_1, J_2) on M_1, and the order (J_2, J_1) on M_2, is the unique optimal solution. ■

We now turn to determining the complexity of $F2|s_{ij}, r_{ij}|C_{max}$ not restricted to permutation solutions. To show this problem to be NP-hard in the strong sense we reduce the well-known 3-PARTITION problem which is NP-complete in the strong sense to the decision counterpart of the problem under consideration.

Define the decision version of $F2|s_{ij}, r_{ij}|C_{max}$ as follows:

Given an instance of $F2|s_{ij}, r_{ij}|C_{max}$ and a positive integer y, does there exist a schedule S with $C_{max}(S) \leq y$?

Presenting 3-PARTITION, we follow Garey and Johnson (1979):

Given a set $T = \{1, 2, \ldots, 3t\}$ with an integer size e_i for each $i \in T$, and given a positive integer E, such that $\sum_{i \in T} e_i = tE$ and $E/4 < e_i < E/2$. Can T be partitioned into t disjoint sets $T_1, T_2, \ldots, T_t$ such that, for $1 \leq k \leq t$, $\sum_{i \in T_k} e_i = E$? (Note that, if 3-PARTITION has a solution, then $|T_k| = 3$ for all $k = 1, 2, \ldots, t$).

Theorem 2.1: The problem $F2|s_{ij}, r_{ij}|C_{max}$ not restricted to permutation solutions is NP-hard in the strong sense.

Proof. We reduce 3-PARTITION problem to the decision version of $F2|s_{ij}, r_{ij}|C_{max}$ and show that 3-PARTITION has a solution if and only if this decision problem has a solution.

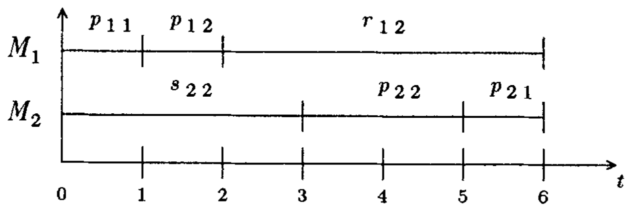

Fig. 2.1

The following instance for the decision version of $F2|s_{ij}, r_{ij}|C_{max}$ is used. There are $n = 4t$ jobs. We divide the jobs into two groups: U-jobs denoted by U_i, $i = 1, 2, \ldots, 3t$, and V-jobs denoted by $V_j, j = 1, 2, \ldots, t$. Their setup, processing, and removal times are given below:

$$s_{1,U_i} = r_{1,U_i} = 0, \quad p_{1,U_i} = e_i \ , \qquad i = 1, 2, \ldots, 3t \ ;$$

$$s_{2,U_i} = r_{2,U_i} = 0, \quad p_{2,U_i} = e_i(E + 2) \ , \qquad i = 1, 2, \ldots, 3t \ ;$$

$$s_{1,V_j} = 0, \quad p_{1,V_j} = E, \quad r_{1,V_j} = E(E + 3) \ , \qquad j = 1, 2, \ldots, t \ ;$$

$$s_{2,V_j} = 2E, \quad p_{2,V_j} = E, \quad r_{2,V_j} = 0 \ , \qquad j = 1, 2, \ldots, t \ .$$

The integer y is set to be $tE(E + 5)$. Without loss of generality we may assume that $E \geq 3$. Suppose that 3-PARTITION has a solution, and $T_1, T_2, \ldots, T_t$ are found subsets of set T. Let $\pi_U(T_k)$ denote an arbitrary permutation of the U-jobs with $i \in T_k$. Consider a schedule S shown in Fig. 2.2.

In this schedule, there is no idle time on each machine, and the jobs are ordered on M_1 according to $(\pi_U(T_1), V_1, \pi_U(T_2), V_2, \ldots, \pi_U(T_t), V_t)$, while on M_2 according to $(V_1, \pi_U(T_1), V_2, \pi_U(T_2), \ldots, V_t, \pi_U(T_t))$. It is easy to check that the length of this schedule is $tE(E + 5)$.

Now suppose that a schedule S exists with length $C_{max}(S) \leq tE(E + 5)$. We prove that 3-PARTITION has a solution. Since the V-jobs are identical, we may assume that they are scheduled on M_2 in increasing order of their numbering.

Since performing all operations on each M_1 and M_2 takes $tE(E + 5)$ time units, we know that $C_{max}(S) = tE(E + 5)$, and that there is no idle time on either machine.

It follows that V_1 must be placed first on M_2, because $s_{2,V_j} > 0$ and $s_{2,U_i} = 0$ while all processing times are strictly positive.

Since $s_{2,V_1} = 2E > E = p_{1,V_1}$, we must consider the possibility that some jobs precede V_1 on M_1. The total length of these jobs on M_1 cannot be greater than

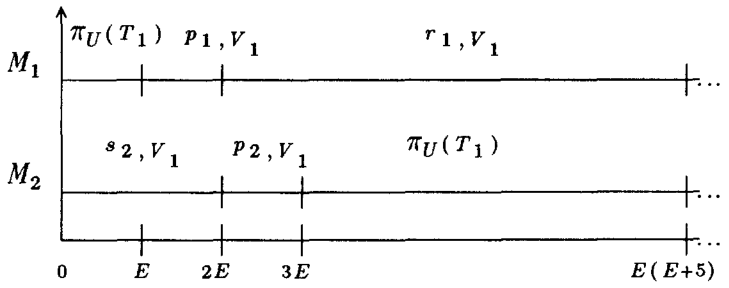

Fig. 2.2

E, because otherwise an idle time arises on M_2 after the completion of the setup of V_1 on M_2. Thus, these jobs can only be U-jobs. We denote the set of indices of the U-jobs preceding V_1 on M_1 by T^*, and the total length of the jobs U_i, $i \in T^*$, on M_1 by E'. Note that $E' \leq E$.

Suppose that $E' < E$. Since all values are integer, we have that $0 \leq E' \leq E - 1$. Since

$$C_{1,V_1}(S) = E' + p_{1,V_1} + r_{1,V_1} = E' + E(E + 4) > 3E = s_{2,V_1} + p_{2,V_1}$$

$$= C_{2,V_1}(S) \ ,$$

we conclude that V_1 must be followed on M_2 by the jobs U_i, $i \in T^*$. However, the last of those jobs finishes at $3E + E'(E + 2)$ which is still less than $C_{1,V_1}(S)$. To avoid an idle time on M_2 we can only start the setup stage of job V_2 on M_2 immediately after the last of the jobs U_i, $i \in T^*$, has been completed. So, the setup stage of O_{2,V_2} finishes at $5E + E'(E + 2)$, and the processing stage of that operation must start exactly at this time. On the other hand, even if V_2 directly follows V_1 on M_1, the processing stage of O_{1,V_2} finishes at $E' + p_{1,V_1} + r_{1,V_1} + p_{1,V_2} = E' + E(E + 5)$. We have that $5E + E'(E + 2) < E' + E(E + 5)$ since this inequality is equivalent to $E'(E + 1) < E^2$, the latter one being true due to $E' \leq E - 1$. Thus, if $E' < E$ one cannot avoid an idle time on M_2, and we conclude that $E' = E$.

We have shown that the total length of the jobs U_i, $i \in T^*$, on M_1 is equal to E, and thus, there must be exactly three of them. We denote $T_1 = T^*$ getting $\sum_{i \in T_1} e_i = E$.

Extending arguments presented above, one can prove that in schedule S with the length $tE(E + 5)$ machine M_1 processes exactly three U-jobs with the total length E during each time interval $[(k - 1)E(E + 5), kE(E + 5)]$, $k = 1, 2, \ldots, t$. Denoting the set of indices of the U-jobs processed on M_1 during interval $[(k - 1)E(E + 5), kE(E + 5)]$ by T_k, $k = 1, 2, \ldots, t$, we obtain a solution of 3-PARTITION. ■

It is interesting to find out whether the best permutation solution can have significantly greater schedule length than the length of a global optimal solution.

We denote the value of the makespan for a global optimal solution and for the best permutation solution by C^*_{max} and C^{π}_{max}, respectively. For a non-empty set of jobs Q, we define $G_1(Q)$ to be $\sum_{J_j \in Q} G_{1,j}$; $G_2(Q)$ is defined analogously. Let $\pi(Q)$ denote an arbitrary permutation of the jobs of a set Q. If $Q = \varnothing$ then $G_1(Q) = G_2(Q) = 0$.

Theorem 2.2: For $F2|s_{ij}, r_{ij}|C_{max}$, the following bound

$$C^{\pi}_{max}/C^{*}_{max} \leq 3/2 \ . \tag{2.3}$$

holds, and this bound is tight.

Proof: The proof is based on the results obtained by Strusevich (1993) for the two machine open shop scheduling problem with setup, processing, the removal times separated.

Let N denote the set of all jobs $J_j, j = 1, 2, \ldots, n$. For each job, find a_j and b_j by formula (2.1), and divide N into two sets:

$$N^a = \{J_j | a_j \leq b_j\} = \{J_j | G_{1,j} \leq G_{2,j}\} \ ,$$

$$N^b = \{J_j | a_j > b_j\} = \{J_j | G_{1,j} > G_{2,j}\} \ .$$

If $N^a \neq \varnothing$, then select a job $J_k \in N^a$ such that $b_k = \max\{b_j | J_j \in N^a\}$, otherwise assume $\{J_k\} = \varnothing$. If $N^b \neq \varnothing$, then select a job $J_l \in N^b$ such that $a_l = \max\{a_j | J_j \in N^b\}$, otherwise assume $\{J_l\} = \varnothing$.

For a schedule S, let $R^p_{ij}(S)$ and $C^p_{ij}(S)$ denote the starting and the completion time of the *processing* phase of operation O_{ij}, $i = 1, 2$, $J_j \in N$. Note that, in any feasible schedule S, the inequality $C^p_{1j}(S) \leq R^p_{2j}(S)$ holds for each job J_j.

Suppose that $|N^a| = q$ and $|N^b| = m$. Determine permutations $\phi(N_A) = (i_1, i_2, \ldots, i_q) = (J_k, \pi(N^a \backslash \{J_k\}))$ and $\psi(N_B) = (f_1, f_2, \ldots, f_m) = (\pi(N^b \backslash \{J_l\}), J_l)$.

Consider a schedule S^a in which each machine processes the jobs of set N^a according to a sequence $\phi(N^a)$, provided that both machines do not stand idle once started.

For schedule S^a, let R^a_2 be the starting time of machine M_2. It follows from (2.2) that

$$R^a_2 = \max\left\{a_{i_1}, 0, \max\left\{\sum_{j=1}^{u} a_{i_j} - \sum_{j=1}^{u-1} b_{i_j} \middle| u = 2, 3, \ldots, q\right\}\right\} \ .$$

However, due to the definitions of set N^a and noticing that $k = i_1$, we obtain for each u, $2 \leq u \leq q$, that

$$\sum_{j=1}^{u} a_{i_j} - \sum_{j=1}^{u-1} b_{i_j} \leq a_k \ .$$

Thus, in this case, we have $R_2 = \max\{a_k, 0\}$. Schedule S^a is shown in Fig. 2.3.

Similarly, we construct a schedule S^b where each machine processes the jobs of set N^b according to the sequence $\psi(N_B)$ provided that both machines do not stand idle once started.

For schedule S^b, let R_2^b be the starting time of machine M_2. It follows from (2.2) that

$$R_2^b = \max\left\{a_{f_1}, 0, \max\left\{\sum_{j=1}^{u} a_{f_j} - \sum_{j=1}^{u-1} b_{f_j} \middle| u = 2, 3, \ldots, m\right\}\right\} .$$

However, due to the definitions of set N^b and noticing that $l = f_m$, we obtain for each u, $2 \le u \le m-1$, that

$$\sum_{j=1}^{m} a_{f_j} - \sum_{j=1}^{m-1} b_{f_j} \ge \sum_{j=1}^{u} a_{f_j} - \sum_{j=1}^{u-1} b_{f_j} .$$

In addition, it follows that $a_l \ge a_{f_1}$ and, since

$$\sum_{j=1}^{m-1} a_{f_j} - \sum_{j=1}^{m-1} b_{f_j} \ge 0 ,$$

we obtain

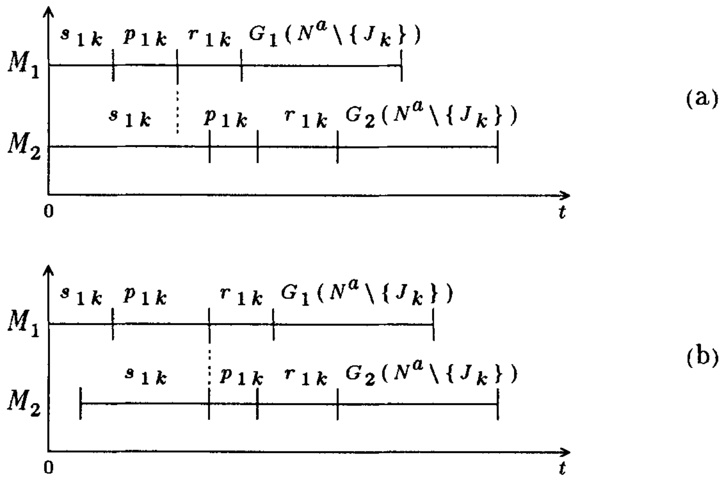

Fig. 2.3

$$\sum_{j=1}^{m} a_{f_j} - \sum_{j=1}^{m-1} b_{f_j} \geq a_{f_1} \ .$$

Finally, we conclude that $R_2^b = \max\left\{\sum_{j=1}^{m} a_{f_j} - \sum_{j=1}^{m-1} b_{f_j}, 0\right\}$. Schedule S^b is shown in Fig. 2.4.

Now schedules S^a and S^b can be combined as shown in Fig. 2.5 to obtain a schedule S associated with a permutation $\pi = (J_k, \pi(N^a \backslash \{J_k\}), \pi(N^b \backslash \{J_l\}), J_l)$. Note that if either $N^a = \emptyset$ or $N^b = \emptyset$, then $\pi = (\pi(N^b \backslash \{J_l\}), J_l)$ or $\pi = (J_k, \pi(N^a \backslash \{J_k\}))$, respectively.

It follows that

$$C_{max}(S) = \max\{G_1(N), G_2(N), a_k + G_2(N), b_l + G_1(N)\} \ . \tag{2.4}$$

If $C_{max}(S) = \max\{G_1(N), G_2(N)\}$ then S is a global optimal schedule, and (2.3) holds.

We show that if $C_{max}(S) = \max\{a_k + G_2(N), b_l + G_1(N)\}$, then $C_{max}(S)/C^*_{max} \leq 3/2$. Thus, since $C^{\pi}_{max} \leq C_{max}(S)$, we obtain bound (2.3).

Suppose that $C_{max}(S) = a_k + G_2(N)$. It is evident, that if $a_k \leq G_2(N)/2$, then $C_{max}(S)/ \leq 3G_2(N)/2$. Since $C^*_{max} \geq G_2(N)$, we have that $C_{max}(S)/C^*_{max} \leq 3/2$.

Suppose that $a_k > G_2(N)/2$. Then, we have that $G_{2,k} \geq b_k \geq a_k > G_2(N)/2$, and it follows that $C_{max}(S) = a_k + G_2(N) < 2a_k + G_{2,k}$.

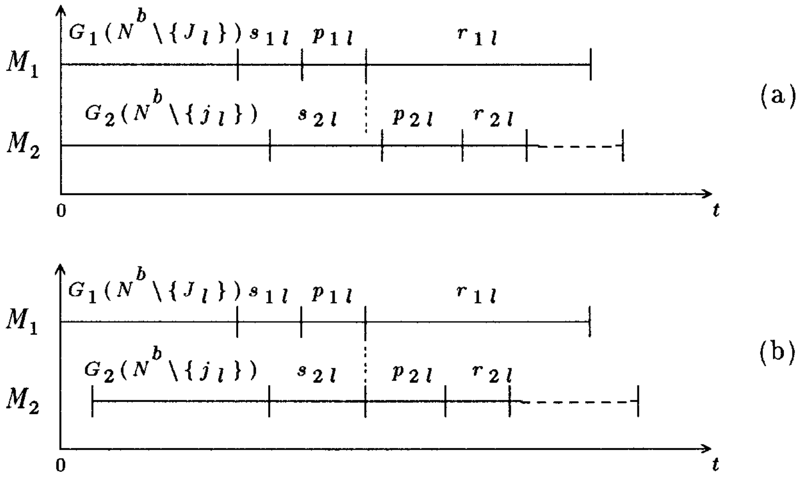

Fig. 2.4

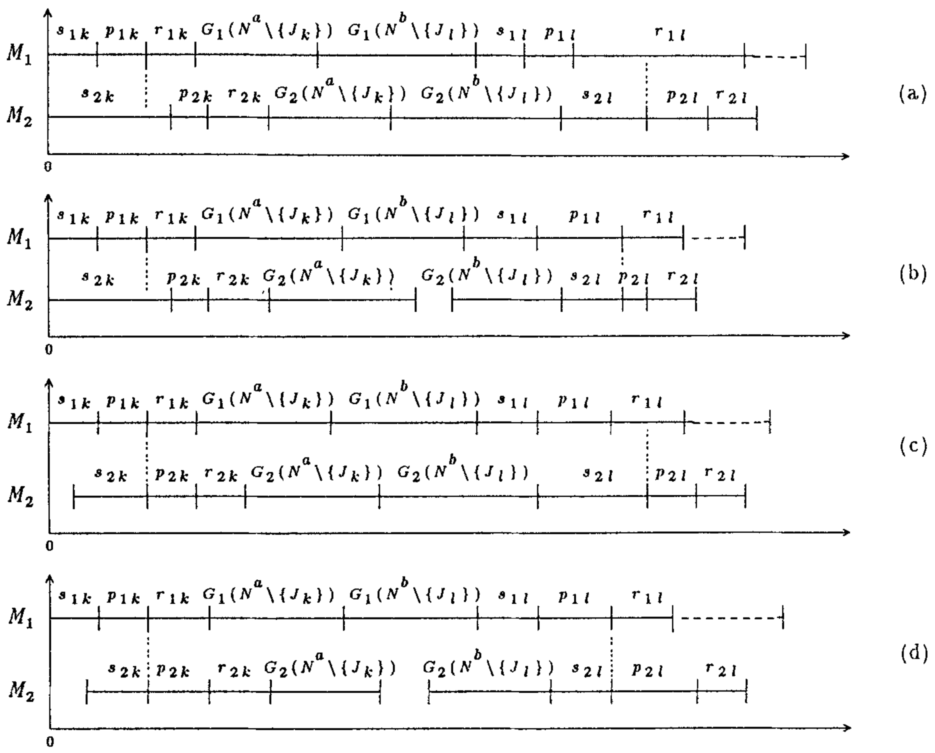
M_1
M_2
s_{1k} p_{1k} r_{1k} $G_1(N^a\setminus\{J_k\})$ $G_1(N^b\setminus\{J_l\})$ s_{1l} p_{1l} r_{1l}
s_{2k} p_{2k} r_{2k} $G_2(N^a\setminus\{J_k\})$ $G_2(N^b\setminus\{J_l\})$ s_{2l} p_{2l} r_{2l}
0
(a)
(b)
(c)
(d)

Fig. 2.5

On the other hand,

$$C^*_{max} \geq s_{1,k} + p_{1,k} + p_{2,k} + r_{2,k} = s_{1,k} + p_{1,k} + (-s_{2,k} + s_{2,k}) + p_{2,k} + r_{2,k}$$
$$= a_k + G_{2,k} \ .$$

Thus, we have that

$$C_{max}(S)/C^*_{max} < (2a_k + G_{2,k})/(a_k + G_{2,k}) = 1 + a_k/(a_k + G_{2,k}) \leq 3/2 \ .$$

For $C_{max}(S) = b_l + G_1(N)$ the proof is similar.

We have proved that bound (2.3) holds. The following example shows that this bound is tight. There are two jobs, J_1 and J_2, with the following setup, processing, and removal times:

J_1: $p_{1,1} = p_{2,1} = 1$, all other times are zero ;

J_2: $r_{1,2} = s_{2,2} = 1$, all other times are zero .

It is easy to check that there is a unique optimal schedule with $C^*_{max} = 2$ (see Fig. 2.6). In this schedule, the job order on M_1 is (J_1, J_2), and on M_2 is (J_2, J_1). On the other hand, each permutation schedule has length 3. ■

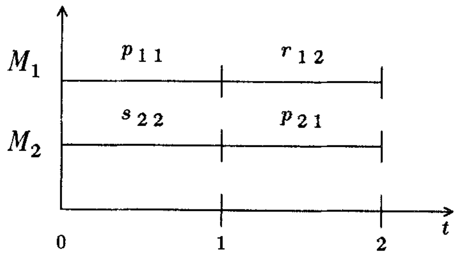

Fig. 2.6

Note that in the proof of Theorem 2.2, we use heuristic schedule S which can be found in $O(n)$ time. This leads to a linear-time approximation algorithm for $F2|s_{ij}, r_{ij}|C_{max}$. Note that the best permutation schedule also can be treated as an approximate solution of $F2|s_{ij}, r_{ij}|C_{max}$, but this schedule can be found in $O(n \log n)$ time while its worst-case ratio bound is the same as for schedule S.

3 The Problem $F2|b|C_{max}$

In this section, we examine $F2|b|C_{max}$ with $1 \leq b \leq n-2$. Recall that if $b \geq n-1$ then $F2|b|C_{max}$ is equivalent to $F2| \,|C_{max}$, while $F2|b=0|C_{max}$ is equivalent to $F2| \,|C_{max}$ with the no-wait restriction. For both $F2| \,|C_{max}$ and its no-wait counterpart one may look for an optimal solution among permutation schedules. A permutation version of $F2|b|C_{max}$ was studied by Papadimitriou and Kanellakis (1980) who restricted themselves to considering the *FIFO* buffer policy. We show that if one accepts an arbitrary buffer policy then an optimal solution need not be a permutation schedule.

Lemma 3.1: For $F2|b|C_{max}$, there may be no optimal solution that is a permutation schedule.

Proof: For any b, $1 \leq b \leq n-2$, we present an instance of $F2|b|C_{max}$ with a unique optimal solution which is not a permutation schedule. Consider the input data shown in Table 3.1.

We define

Job order on M_1: type 1, b times type 2, type 3, $(b+2)$ times type 4 ;

Job order on M_2: type 1, b times type 2, 1 time type 4, type 3, $(b+1)$ times type 4 ;

Table 3.1

Type of jobs	$p_{1,j}$	$p_{2,j}$	Number of jobs of this type
1	1	$5b^2 + 2$	1
2	2	$3b + 3$	b
3	$8b^2$	$(b+1)^2$	1
4	$b + 2$	1	$b + 2$

and consider a schedule with each operation starting as early as possible. See Fig. 3.1.

We show that this is the only optimal schedule. In the proof, we denote by $O_{i,2}(O_{i,4})$ an operation of any job of type 2 (of type 4, respectively) on M_i, $i = 1, 2$.

Note that the above schedule does not contain any idle time, except the minimal release time for M_2 and minimal idle on M_1 after all jobs have been completed on that machine. Thus, the job of type 1 should be placed first on both M_1 and M_2, and one of the jobs of type 4 must be placed last on M_2.

Scheduling $O_{1,3}$ in the second position on M_1 induces an idle time on M_2 since $8b^2 > 5b^2 + 2$. Thus, we have to fill up the time gap between $O_{1,1}$ and $O_{1,3}$.

There may be no more than b jobs between $O_{1,1}$ and $O_{1,3}$, otherwise it would lead to an idle time on M_1 due to the buffer constraint.

Suppose that we fill up the gap between $O_{1,1}$ and $O_{1,3}$ on M_1 with k times $O_{1,2}$ and l times $O_{1,4}$, $l + k \leq b$, and place the corresponding operations on M_2 after $O_{2,1}$. Then $O_{1,3}$ finishes at $1 + 2k + l(b + 2) + 8b^2$. The last of inserted jobs finishes on M_2 at $1 + (5b^2 + 2) + k(3b + 3) + l$. We need the inequality

$$2k + l(b + 2) + 8b^2 \leq (5b^2 + 2) + k(3b + 3) + l$$

which is only valid if $k = b$ and $l = 0$.

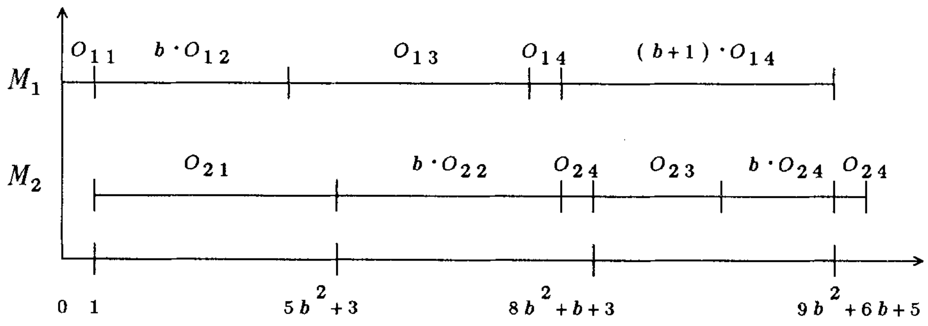

Fig. 3.1

Thus, all b jobs of type 2 must be placed on M_1 before $O_{1,3}$, while all $b + 2$ jobs of type 4 are scheduled after that operation. Besides, all jobs of type 2 must be processed on M_2 after $O_{2,1}$. Note that the first of jobs of type 4 finishes on M_1 at the same time as the last of operations $O_{2,2}$ is completed since $2b + 8b^2 + (b + 2) = (5b^2 + 2) + b(3b + 3)$.

We still need to specify the order of jobs on M_2 after the last operation $O_{2,2}$. Suppose that $O_{2,3}$ is assigned right after that operation. Note that by the time $O_{2,3}$ starts, the first of the operations $O_{1,4}$ finishes and goes to the buffer. Thus, while $O_{2,3}$ is being processed no more than $b - 1$ operations $O_{1,4}$ can be processed on M_1 due to the buffer restriction. However, their total length is $(b - 1)(b + 2) < (b + 1)^2$, and two remaining operations $O_{1,4}$ can only start after some idle time on M_1.

It is obvious that there may be at most one operation $O_{2,4}$ between the last of the operations $O_{2,2}$ and operation $O_{2,3}$ because processing a job of type 4 takes more time on M_1 than on M_2.

Thus, we assign exactly one operation $O_{2,4}$ directly after b operations $O_{2,2}$.

The moment that $O_{2,3}$ starts processing on M_2, the buffer becomes empty. The remaining $b + 1$ operations $O_{2,4}$ can be processed without any idle time, because $b(b + 2) < 1 + (b + 1)^2$, and $(b + 1)(b + 2) = 1 + (b + 1)^2 + b$. ■

Papadimitriou and Kanellakis (1980) showed that $F2|b|C_{max}$, restricted to permutation schedules, in NP-hard in the strong sense, via a transformation from Numerical $(b + 2)$-Dimensional Matching. Howevr, their restriction to permutation schedules is not crucial, because the instance used has a unique optimal solution that is a permutation schedule. These observations are sufficient for deriving the following result.

Theorem 3.1: The problem $F2|b|C_{max}$ is NP-hard in the strong sense.

For $F2|b|C_{max}$, we denote the makespan for a global optimal solution and for the best permutation solution by $C^*_{max}(b)$ and by $C^\pi_{max}(b)$, respectively. Papadimitriou and Kanellakis showed that

$$C^*_{max}(b = 0)/C^\pi_{max}(b) \leq (2b + 1)/(b + 1) \ , \tag{3.1}$$

and this bound is tight. From (3.1), Papadimitriou and Kanellakis derived an $O(n \log n)$ approximation algorithm for $F2|b|C_{max}$ restricted to permutation solutions: use the Gilmore-Gomory algorithm (see Gilmore and Gomory, 1964) to solve $F2|b = 0|C_{max}$, and schedule the jobs according to the resulting permutation, using the buffer of capacity b. This algorithm turns out to have the tight worst-case bound $(2b + 1)/(b + 1)$ as well. Note that for $F2|b = 0|C_{max}$ only permutation schedules are feasible.

It easy to check that the part of the proof presented by Papadimitriou and Kanellakis (1980) to show that $(2b + 1)/(b + 1)$ is an upper bound on the ratio $C^*_{max}(b = 0)/C^{\pi}_{max}(b)$ can be applied to $F2|b|C_{max}$ in its general form, because no one of the arguments used is restricted to permutation solutions. So, we even have that

$$C^*_{max}(b = 0)/C^*_{max}(b) \leq (2b + 1)/(b + 1) \ . \tag{3.2}$$

Note that the example given by Papadimitriou and Kanellakis shows that this bound is tight. It follows from (3.2) that the heuristic solution based on the Gilmore-Gomory algorithm is as good for $F2|b|C_{max}$ in the general form as for its permutation counterpart.

Acknowledgements: The authors want to thank Stan van Hoesel from Eindhoven University of Technology, The Netherlands, for his help on the proof of Lemma 3.1. Comments from the anonymous referees have contributed to improving the presentation.

References

Conway RW, Maxwell WL, Miller LW (1967) Theory of scheduling. Addison-Wesley, Reading

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of $\mathcal{NP}$-completeness, Freeman, San Francisco

Garey MR, Johnson DS, Sethi R (1976) The complexity of flow shop and job shop scheduling. Mathematics of Operations Research 1:117–129

Gilmore PC, Gomory RE (1964) Sequencing a one-state variable machine: a solvable case of the traveling salesman problem. Operations Research 12:665–679

Johnson SM (1954) Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1:61–68

Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (1993) Sequencing and scheduling: algorithms and complexity. In: Handbook in operations research and management Science, Vol. 4: Logistics of production and inventory (Graves SC et al. eds), North-Holland 445–522

Papadimitriou CH, Kanellakis PC (1980) Flowshop scheduling with limited temporary storage. Journal of the Association for Computing Machinery 27:533–549

Strusevich VA (1993) Two machine open shop scheduling problem with setup, processing and removal times separated. Computers and Operations Research 20:597–611

Sule DR (1982) Sequencing *n* jobs on two machines with setup, processing and removal times separated. Naval Research Logistics Quarterly 29:517–519

Sule DR and Huang KY (1983) Sequencing on two and three machines with setup, processing and removal times separated. International Journal of Production Research 21:723–732

Yoshida T and Hitomi K (1979) Optimal two-stage production scheduling with setup times separated. AIIE Transactions 11:261–263

Received: November 1991
Revised version received: March 1993