Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

ELSEVIER

Computers & Operations Research 32 (2005) 599–611

computers & operations research

www.elsevier.com/locate/dsw

Total completion time minimization in a computer system with a server and two parallel processors

S. Guirchoun[a,b,*], P. Martineau[a], J.-C. Billaut[a]

[a] *Laboratoire d'Informatique, Ecole Polytechnique de l'Université de Tours, 64 av J Portalis, 37200 Tours, France*
[b] *Accellent, 27 rue Michael Faraday, 37170 Chambray les Tours, France*

Abstract

The context of the problem tackled in this paper is a computer system composed by a single server and two identical parallel machines. The processing times on the server are assumed to be unary and the objective is to minimize the total completion time. The papers dealing with scheduling problems with a server generally consider that the setup activities require simultaneously the server and the machine. In this paper, this constraint is not considered and the studied problem is a two-stage hybrid flow shop with no-wait constraint between the two stages. An algorithm that can solve optimally this problem in $O(n \log(n))$ time is proposed. Finally, it is shown that solving this problem optimally leads to an optimal solution to the problem without the no-wait constraint.
© 2003 Elsevier Ltd. All rights reserved.

Keywords: Scheduling; Parallel machines; Single server; Hybrid flow shop

1. Introduction

Parallel processing has received a lot of attention these last years because of its efficiency, which is a crucial part in success of parallel computer systems. Hence, there is a necessity for developing efficient scheduling algorithms in computer systems. A classical configuration is build with one server of files linked with clients by a network [1]. The first step of the parallelization consists in dispatching on a designed client the code and the data of each program that can be parallelized. For each program that has to be sent to a client, the server must, successively, read the code and data on the hard disk and send them to the client by the network. In a first approximation these two steps can be modeled by one task on the server. This assumption is not excessive because in

* Corresponding author. Tel.: +33-247-361-414; fax: +33-247-361-422.
E-mail addresses: samuel.guirchoun@etu.univ-tours.fr (S. Guirchoun), pmartineau@univ-tours.fr (P. Martineau), billaut@univ-tours.fr (J.-C. Billaut).

0305-0548/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2003.08.007

such a context, the designer tries to balance the processing times on different clients to obtain the best performance of the parallelized execution. Then programs and data have often similar sizes and as a consequence, reading and sending them takes similar times. The goal to achieve can be the minimization of the maximum completion time if the user is waiting for fast result. But if the goal is to use the network with the most efficiency, the objective that has to be studied is the minimization of the mean completion time or the sum of completion times.

We consider a deterministic scheduling environment with m identical parallel machines $M_1, M_2, \dots, M_m$ and n jobs to schedule. Each job must be processed without preemption on a machine to determine. Before its processing, a job has to be loaded on a machine. This loading activity or setup activity, is performed by another special machine, called a server. After a setup, the server is available again to perform another loading activity, that is to say a single server can handle only one job at a time and can be considered like a single machine. The loading of a job must be immediately followed by its processing. In this environment, we consider that setup times require a unit time for any job. Moreover, we assume that the transfer times between the server and the machines are non-significant. The aim of this paper is to find a feasible schedule in an environment of $m = 2$ machines, and with minimum total completion time, which corresponds to the minimization of the work in process in the network.

Many results of the last few years are issued from parallel machines scheduling problems with server. Koulamas [2] proposes a beam search heuristic algorithm for a static environment with two parallel processors and a single server where the aim is to find a feasible schedule which minimizes the machine idle time resulting from the unavailability of the server. Kravchenko and Werner [3], Hall et al. [4] and Brucker et al. [5] present a lot of complexity results for these problems. Glass et al. [6] consider related models with parallel machines for which jobs are dedicated and provide algorithmic, complexity and heuristic analysis results. Kravchenko and Werner [7] propose a heuristic to minimize the sum of the completion times in the case of unit setup times and arbitrary processing times. However, the problems tackled in these papers implicitly consider production environment where the server can be a human operator, a robot or an automated guided vehicle. Hence, during the loading operation, the performing machine cannot process another job. So, the loading activity is usually considered like a multiprocessor task, that requires simultaneously the server and the machine to be performed.

In a computer system, the server that send data to machines is called a network server. During the loading activity, it is not necessary for the performing machine to be available: it can process another job. Indeed, machines have a communication coprocessor which allows them to receive server information at any time. So, in a computer system, the loading activity can be considered like a job that requires only the server to be performed.

The computer system that we consider can be seen as a two-stage hybrid flow shop or multi-processor flow shop, with a single machine at the first stage, which is the server and m parallel machines at the second stage, with a no-wait constraint between the two stages. These problems are known to be strongly NP-hard [8] even for their preemptive version [9]. Vignier et al. [10] and Linn and Zhang [11] propose a state-of-the-art survey on hybrid flow shop scheduling problems. However, most of the problems consider the makespan as criterion. In the case of the minimization of the sum of completion times, Pinedo [12] shows that it is possible to find an optimal solution using SPT rule, when all the operations of a job have the same processing times. Unfortunately, this result is no longer valid under the hypotheses that we consider.

In Section 2, the notations are introduced and a mathematical formulation is given for the general problem (m machines). In Section 3, complexity results and resolution methods are given for two particular cases where $m = 2$: the processing times at the second stage are all less than 1 and the processing times at the second stage are all strictly greater than 1. In Section 4, we present a polynomial reduction between parallel machine problems with server and hybrid flow shop problems. In Section 5, a polynomial time algorithm is proposed to solve optimally the case where $m=2$, with integer processing times. Finally, in Section 6 we show that this algorithm solves also optimally the case without no-wait constraint.

2. Notations and mathematical formulation

2.1. Notations

According to the notation introduced in [13] and extended in [10], the hybrid flow shop problem under consideration is denoted by $FH2,(1,Pm)|nowait, p_{i,1} = 1|\sum C_i$; the first field α indicates a two-stage hybrid flow shop with a single machine at the first stage, m identical parallel processors at the second stage. Fields β and γ are the classical ones in scheduling literature.

We consider a set J of n jobs $\{i\}_{1 \leqslant i \leqslant n}$ to schedule on two stages. Each job $i (1 \leqslant i \leqslant n)$ is composed by two operations: the operation $o_{i,1}$ processed at the first stage and the operation $o_{i,2}$ processed at the second stage. We assume that preemption is not allowed and that each machine can process only one operation at a time. We denote by $p_{i,1}$ and $p_{i,2}$ the processing times of operation $o_{i,1}$ and operation $o_{i,2}$, respectively; we assume that $p_{i,1}=1, \forall i, 1 \leqslant i \leqslant n$; $p_{[\ell],j}$ is the processing time at stage j of the operation in position ℓ at the first stage ($1 \leqslant j \leqslant 2, 1 \leqslant \ell \leqslant n$). In this paper, we consider that the processing times at the second stage are positive integers. C_i refers to the completion time of job i and $C_{i,j}$ refers to the completion time of operation $o_{i,j}$ with $1 \leqslant j \leqslant 2 (C_{i,2} = C_i)$. We denote by $T_k (1 \leqslant k \leqslant m)$ the date before which it is not possible to perform a job on machine M_k at the second stage. Because the processing times of the operations are unary at the first stage, we have $T_k = k, \forall k$.

Shortest processing time (SPT) is a non-decreasing order of the second stage processing times rule and first available machine (FAM) schedules the current operation on the first available machine. SPT/FAM denotes an $O(n \log(n))$ resolution method [4] that sorts the jobs according to SPT rule in $O(n \log(n))$ time and then assign them according to FAM rule in $O(n)$ time.

The notation $Pm, S|s_i|\gamma$ introduced by Hall et al. [4] represents the scheduling problems with a server S and m parallel machines. As mentioned below, in these problems, setup s_i of job i must be processed by the server and a parallel machine. In the following, we denote by *PS-problems* the parallel machine problems with a single server under this constraint and *FH2-problems* the two-stage hybrid flow shop problems for which this constraint is not valid.

2.2. Integer linear programming model

The data of the problem are: n, m and $p_{i,j}, 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant 2$. H is an arbitrary high value. The variables are: $t_{i,j}$, the starting time of operation $o_{i,j}, 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant 2$; $x_{i,k}$ equals to 1 if operation $o_{i,2}$ is assigned to machine M_k and 0 otherwise, $1 \leqslant i \leqslant n, 1 \leqslant k \leqslant m$; $y_{i,\ell,j}$ equals to 1 if job i precedes job ℓ at stage $j, 1 \leqslant i \leqslant n, 1 \leqslant \ell \leqslant n, i \neq \ell, 1 \leqslant j \leqslant 2$. The objective function is to minimize $\sum_{i=1}^{n}(t_{i,2}+p_{i,2})$.

$$\text{Minimize} \quad \sum_{i=1}^{n}(t_{i,2}+p_{i,2}) \tag{1}$$

$$\text{subject to} \quad \sum_{k=1}^{m} x_{i,k}=1, \quad i=1,\dots,n, \tag{2}$$

$$t_{i,2}=t_{i,1}+p_{i,1}, \quad i=1,\dots,n, \tag{3}$$

$$t_{\ell,1} \geqslant t_{i,1}+p_{i,1}-H\times(1-y_{i,\ell,1}), \quad i=1,\dots,n,\ \ell=1,\dots,n,\ i\neq\ell, \tag{4}$$

$$t_{i,1} \geqslant t_{\ell,1}+p_{\ell,1}-H\times(y_{i,\ell,1}), \quad i=1,\dots,n,\ \ell=1,\dots,n,\ i\neq\ell, \tag{5}$$

$$t_{\ell,2} \geqslant t_{i,2}+p_{i,2}-H\times(1-y_{i,\ell,2})-H\times(2-x_{i,k}-x_{\ell,k}),$$
$$i=1,\dots,n,\ \ell=1,\dots,n,\ i\neq\ell,\ k=1,\dots,m, \tag{6}$$

$$t_{i,2} \geqslant t_{\ell,2}+p_{\ell,2}-H\times(y_{i,\ell,2})-H\times(2-x_{i,k}-x_{\ell,k}),$$
$$i=1,\dots,n,\ \ell=1,\dots,n,\ i\neq\ell,\ k=1,\dots,m, \tag{7}$$

$$t_{i,j} \geqslant 0, \quad i=1,\dots,n,\ j=1,2, \tag{8}$$

$$x_{i,k}\in\{0,1\},\ i=1,\dots,n,\ k=1,\dots,m, \tag{9}$$

$$y_{i,\ell,j}\in\{0,1\},\ i=1,\dots,n,\ \ell=1,\dots,n,\ i\neq\ell,\ j=1,2. \tag{10}$$

Constraints (2) ensure that each job is assigned to exactly one machine at the second stage. The routing constraints and the no-wait constraint are formulated in (3). Constraints (4) and (5) (respectively (6) and (7)) are the disjunctive constraints at the first stage (respectively at the second stage). This model contains $nm+2n(n-1)$ boolean variables and $2n$ positive variables and $4n+(m+2)n(n-1)$ constraints.

3. Particular cases

The problem under consideration is denoted by $FH2,(1,P2)|nowait,p_{i,1}=1|\sum C_i$. For the $P2,S|s_i=1|\sum C_i$ problem, Hall et al. [4] show that the SPT/FAM list algorithm returns an optimal solution. Unfortunately, when the multiprocessor constraint is released, i.e. when the loading activity can be performed on the server, without the need of the performing machine, this algorithm is not optimal. We detail in this section two particular cases of the problem $FH2,(1,P2)|nowait, p_{i,1} = 1|\sum C_i$: $p_{i,2} \leqslant 1$, $\forall i, 1 \leqslant i \leqslant n$ and $p_{i,2} > 1, \forall i, 1 \leqslant i \leqslant n$ and we show the optimality of algorithm SPT/FAM for both cases. For the two following cases, we note that results remain valid when the processing times at the second stage are real numbers.

3.1. Problem with $p_{i,2} \leqslant 1, \forall i,\ i = 1,\dots,n$

If we consider the problem with $p_{i,2} \leqslant 1$ $\forall i, 1 \leqslant i \leqslant n$, we have $\forall i, 1 \leqslant i \leqslant n$ $C_i = C_{i,1} + p_{i,2}$. Thus $\sum_{i=1}^{n} C_{i,2}$ is equivalent to $\sum_{i=1}^{n} C_{i,1}$, which is a constant because no idle time is introduced on the machine at the first stage. This constant is equal to $n(n+1)/2$. Thus, any semi-active schedule is an optimal solution.

3.2. Problem with $p_{i,2} > 1, \forall i,\ i = 1,\dots,n$

We consider that $p_{i,2} > 1$ for all jobs and we propose a list algorithm based on SPT/FAM rule. φ denotes the sequence SPT.

Theorem 1. *Problem $FH2,(1,P2)|nowait, p_{i,1} = 1, p_{i,2} > 1|\sum C_i$ can be solved optimally in $O(n\log(n))$ time by SPT/FAM algorithm.*

Proof. We have seen that the second operation of the first job cannot start before time 1 and of the second job before time 2. Hence, without loss of generality, we set $T_1 = 1$ and $T_2 = 2$. We therefore consider the problem as a two-parallel-machine scheduling problem where M_1 is available at time $T_1 = 1$ and M_2 at time $T_2 = 2$.

M_1 is the FAM, so $\varphi(1)$ is assigned to M_1. Because $p_{\varphi(1),2} > 1$, M_2 becomes the FAM. Then, $\varphi(2)$ is assigned to M_2 and because $p_{\varphi(1),2} \leqslant p_{\varphi(2),2}$ and $T_1 < T_2$, M_1 becomes the FAM. And so on, jobs are assigned alternatively on M_1 and M_2. It follows that the completion times of any two jobs cannot be equal, and because $p_{i,2} > 1$ $\forall i, 1 \leqslant i \leqslant n$, the no-wait constraint with the first stage cannot lead to an idle time on M_1 or on M_2. Consequently, the solution returned by SPT/FAM at the second stage is equivalent to the optimal solution of the $P2||\sum C_i$ problem where machine M_1 is available at time $T_1 = 1$ and M_2 at time $T_2 = 2$ [14]. Thus, it also generates an optimal schedule for problem $FH2,(1,P2)|nowait, p_{i,1} = 1, p_{i,2} > 1|\sum C_i$. □

Corollary 2. *Theorem* 1 *cannot be applied when at least one operation has a processing time at the second stage smaller than or equal to one.*

Proof. The proof of Theorem 1 does not hold if one job is such that $p_{i,2} \leqslant 1$ because in this case, idle times can be introduced on M_1 and M_2 and the result is then no more valid. In Fig. 1, we present an example where SPT/FAM does not return the optimal solution. □

i	$p_{i,1}$	$p_{i,2}$
1	1	1
2	1	1
3	1	2
4	1	3
5	1	4
6	1	6

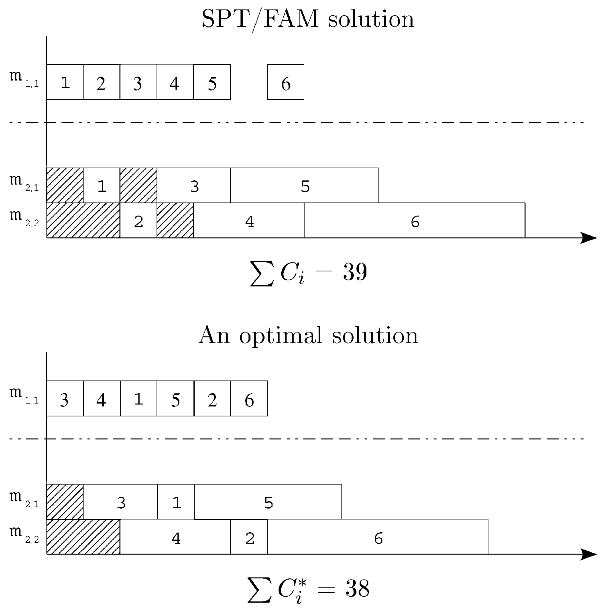

Fig. 1. Example where SPT/FAM is not optimal.

4. Reduction

In this section, we present a polynomial reduction that shows that a two-stage hybrid flow shop problem with a single machine at the first stage, m parallel machines at the second stage, a no-wait constraint between the two stages and equal processing times at the first stage, is at least as difficult as the m parallel machine problem with a single server and equal setup times. This reduction is applicable to the total completion time criterion.

Theorem 3. *For $\alpha \in \{m, \circ\}$, setup times s_i all equal and arbitrary processing times p_i, the problem $P\alpha, S1|s_i = s|\sum C_i$ reduces polynomially to $FH2, (1, P\alpha)|p_{i,1} = s, nowait|\sum C_i$.*

Proof. Consider the following decision problems $P\alpha$ and $FH2$.

Problem: $P\alpha$

Instance: n jobs, $\{p_i\}_{1 \leqslant i \leqslant n}$ the processing times, s and B two positive integers.

Question: Can we find a feasible schedule to the $P\alpha$, $S1|s_i = s|-$ problem with a total completion time less than or equal to B?

Problem: $FH2$

Instance: n' jobs, $p_{i,1} = p$, $\forall i, 1 \leqslant i \leqslant n'$ and $\{p_{i,2}\}_{1 \leqslant i \leqslant n'}$ the processing times at the second stage and a positive integer B'.

Question: Can we find a feasible schedule to the $FH2, (1, P\alpha)|p_{i,1} = p, nowait|-$ problem with a total completion time less than or equal to B'?

It is clear that the decision problem $FH2 \in \mathcal{NP}$ because we can check a positive answer in polynomial time.

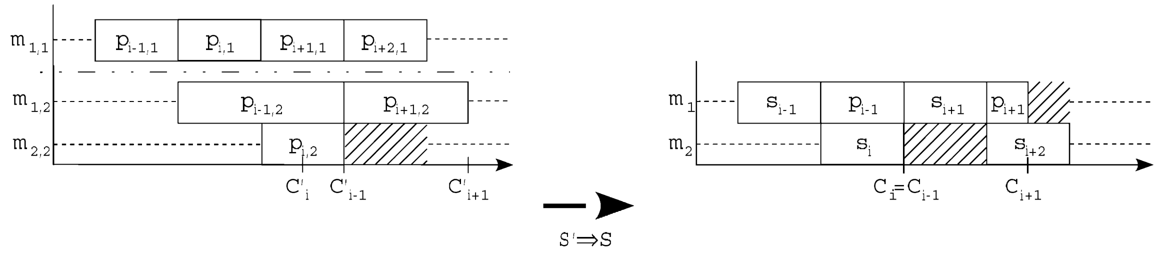

Fig. 2. Schedule S constructed from schedule S'.

We construct an instance I of the problem $FH2$ by the following polynomial transformation: $n' = n$; $p_{i,1} = p = s$, $\forall i, 1 \leqslant i \leqslant n$; $p_{i,2} = p_i + s_i$, $\forall i$, $1 \leqslant i \leqslant n$; $B' = B + n * s$.

Suppose that the answer to $P\alpha$ is 'yes', and let S be a feasible schedule for the problem $P\alpha$. Then we have $\sum_{i=1}^{n} C_i(S) \leqslant B$. We build from S a feasible solution to $FH2$ as follows: the setup activities of $P\alpha$ are assigned to the machine at the first stage, and the assignment of the operations to the machines is not changed. This solution is called S', and C'_i denotes the completion time of job i in S'. In S, the difference between the starting times of an operation j and its preceding operation i on a machine is always greater than or equal to $p_i + s$. Then, even with the no-wait constraint, it is possible in S' to assign to the operations at the first stage, the starting times of the corresponding setup activities in S. Then, because of the no-wait constraint, we have $C'_i = C_i + s$, $\forall i$, $1 \leqslant i \leqslant n$. Thus $\sum_{i=1}^{n} C'_i = \sum_{i=1}^{n} C_i + ns$.

Because $\sum_{i=1}^{n} C_i(S) \leqslant B$, $\sum_{i=1}^{n} C'_i(S') \leqslant B + ns \Rightarrow \sum_{i=1}^{n} C'_i(S') \leqslant B'$ and the answer to $FH2$ is 'yes'.

Suppose now that the answer to $FH2$ is 'yes', and denote by S' a feasible solution to the problem $FH2$. Then we have $\sum_{i=1}^{n} C'_i(S') \leqslant B'$ for this solution S'. Because $p_{i,2} = p_i + s$, $\forall i$, $1 \leqslant i \leqslant n$ in S', the difference between the starting time of an operation j and its preceding operation i on a machine is always greater than or equal to $p_i + s$. Thus in S, after reducing the processing time of i to p_i, it is always possible to introduce a setup activity with processing time s before j. Thus in S, the starting times of the operations are the same as in S', and then $C_i = C'_i - s$, $\forall i$, $1 \leqslant i \leqslant n$ (see Fig. 2).

$\sum_{i=1}^{n} C'_i \leqslant B' \Rightarrow \sum_{i=1}^{n} C_i + ns \leqslant B + ns \Rightarrow \sum_{i=1}^{n} C_i \leqslant B$ and then the answer to $P\alpha$ is 'yes'.

So, the problem $P\alpha$ reduces polynomially to problem $FH2$: $P\alpha \propto FH2$. $\square$

So, because the $P2, S1|s_i = s|\sum C_i$ problem is NP-hard [4], we deduce that the $FH2, (1, P2)|p_{i,1} = p, nowait|\sum C_i$ problem is NP-hard. Moreover, solving the $FH2, (1, P2)|p_{i,1} = p, nowait|\sum C_i$ problem with $p_{i,2}$ real numbers is equivalent to solve the $FH2, (1, P2)|p_{i,1} = p, nowait|\sum C_i$ where $p_{i,2}$ are positive integers. Because this problem is NP-hard, the $FH2, (1, P2)|p_{i,1} = 1, nowait|\sum C_i$ problem with $p_{i,2}$ real numbers is NP-hard.

5. Polynomial algorithm

We consider Algorithm 1 presented in Table 1. This algorithm splits the jobs in three sets and sorts one of them according to SPT rule in $O(n \log(n))$ time and then assigns the jobs according

Table 1
Algorithm 1

```
1.  $\Gamma_1 = 1$, $\Gamma_2 = 2$.
2.  Determine sets $A, B$ and $C$.
3.  While ($A \cup B \cup C \neq \emptyset$) Do
4.    Let $M_k$ be the machine such that $\Gamma_k = \min\{\Gamma_1, \Gamma_2\}$.
5.    If ($C \neq \emptyset$) then
6.      Schedule a job $i$ of $C$ on $M_k$.
7.      $C = C \setminus \{i\}$.
8.    Else
9.      If (($|\Gamma_1 - \Gamma_2| = 1$ and $B \neq \emptyset$) or $A \neq \emptyset$) then
10.       Schedule the first job $i$ of $B$ on $M_k$.
11.       $B = B \setminus \{i\}$.
12.     Else
13.       Schedule a job $i$ of $A$ on $M_k$.
14.       $A = A \setminus \{i\}$.
15.     End If
16.   End If
17.   $\Gamma_k = \Gamma_k + p_{i,2}$.
18. EndWhile.
```

to FAM rule in $O(n)$ time. Thus, its complexity is in $O(n \log(n))$ time. Γ_s, Γ_1 and Γ_2 refer to the completion time of the last job on the server, on M_1 and M_2, respectively. $(\Gamma_s, \Gamma_1, \Gamma_2)$ is called the profile of the partial schedule.

We define sets A, C and list B by $A = \{i | p_{i,2} = 1\}, C = \{i | p_{i,2} = 2\}$ and $B = \{i | p_{i,2} > 2\}$, where B is sorted in non decreasing $p_{i,2}$ order. The idea is to place judiciously the jobs with $p_{i,2} < 2$ in order to fill in idle times at the first stage, created by the longest $p_{i,2}$. Doing like this iteratively reduces the total completion time. Indeed, $\sum_{i=1}^{n} C_{i,2} = \sum_{i=1}^{n} C_{i,1} + \sum_{i=1}^{n} p_{i,2}$ because of the no-wait constraint and $\sum_{i=1}^{n} C_{i,1} = n(n+1)/2 + \sum_{i=1}^{n} (n-i+1)\delta_i$, where δ_i is the idle time between job $i-1$ and job i on the server at the first stage. Thus minimizing $\sum_{i=1}^{n} C_{i,2}$ is equivalent to minimizing $\sum_{i=1}^{n} (n-i+1)\delta_i$.

In the following, we denote by $\mathscr{C}$ the list of completion times of jobs at the second stage, sorted in non-decreasing order: $\mathscr{C}_{[\ell]}$ denotes the ℓth completion time. By extension, the job in position ℓ in $\mathscr{C}$ denotes the job with the ℓth completion time in $\mathscr{C}$.

Lemma 4. *For any schedule, we have*

$$\forall k,\ 2 \leqslant k \leqslant n : T_1 + T_2 + \sum_{i=1}^{k} p_{[i],2} \leqslant \mathscr{C}_{[k-1]} + \mathscr{C}_{[k]},$$

where $p_{[i],2}$ refers to the processing time at the second stage of the job in position i in $\mathscr{C}$.

Proof. Suppose, without loss of generality, that the job in position k in $\mathscr{C}$ is assigned to the first machine. Let ℓ be the last job that completes before $\mathscr{C}_{[k]}$ on the second machine.

If there is no such job ℓ, the first job which is assigned to M_2 completes after the job in position k in $\mathscr{C}$, and this job has a position greater than k in $\mathscr{C}$. Thus, $\mathscr{C}_{[k]} = T_1 + \sum_{i=1}^{k} p_{[i],2}$ and $\mathscr{C}_{[k-1]} \geqslant T_2$

and $\mathscr{C}_\ell \geqslant T_2$. Indeed, we have seen that the second operation of the second job cannot start before time $T_2 = 2$, because the processing times of the operations are unary at the first stage. Then $T_1 + T_2 + \sum_{i=1}^{k} p_{[i],2} \leqslant \mathscr{C}_{[k-1]} + \mathscr{C}_{[k]}$.

Otherwise, all jobs in position $1,\dots,k$ in $\mathscr{C}$ fit on the two machines between T_1 and $\mathscr{C}_{[k]}$ on M_1 and T_2 and $\mathscr{C}_\ell$ on M_2. But we have $\mathscr{C}_{[k-1]} \geqslant \mathscr{C}_\ell$ due to the definition of $\mathscr{C}$. Then $\mathscr{C}_{[k]} \geqslant T_1 + \sum_{i\in\varphi_1} p_{[i],2}$ where $\varphi_1 = \{i \in \mathscr{C} | o_{[i],2} \text{ is processed on } M_1\}$. In the same way, $\mathscr{C}_\ell \geqslant T_2 + \sum_{i\in\varphi_2} p_{[i],2}$ where $\varphi_2 = \{i \in \mathscr{C} | o_{[i],2} \text{ is processed on } M_2\}$. Then $T_1 + T_2 + \sum_{i=1}^{k} p_{[i],2} \leqslant \mathscr{C}_{[k-1]} + \mathscr{C}_{[k]}$. □

Theorem 5. *Algorithm 1 returns an optimal solution to the $FH2,(1,P2)|nowait, p_{i,1} = 1|\sum C_i$ problem.*

Proof. In set B, jobs are sorted in nondecreasing order of their processing times at the second stage. Let us define $\alpha = |\{i | p_{i,2} = 2\}|$.

According to Algorithm 1, during the first α steps, all jobs i in C with $p_{i,2} = 2$ are scheduled on M_1 and on M_2 alternatively. So, at an arbitrary step $v \leqslant \alpha$, the partial schedule has a profile $(v, v+1, v+2)$ or $(v, v+2, v+1)$ and the server (at the first stage) performs jobs without idle time in $[0, v]$.

After step α, jobs in A are not scheduled. At step $(\alpha+1)$, Algorithm 1 schedules the first job ℓ of B with $p_{\ell,2} > 2$, in the time interval $[\alpha+1, \alpha+1+p_{\ell,2}]$. At step $(\alpha+2)$, Algorithm 1 schedules a job of A in $[\alpha+2, \alpha+3]$ and iterates with jobs of A until the partial schedule has a profile of type $(\varepsilon, \varepsilon+1, \varepsilon+2)$ or $(\varepsilon, \varepsilon+2, \varepsilon+1)$ with $\varepsilon = \alpha + p_{\ell,2} - 1$. Then, Algorithm 1 schedules the next job of B and the process iterates until set A or set B is empty. Suppose that set B is empty. Then, Algorithm 1 schedules jobs of A consecutively on the same machine. Because there is no idle time on the server and because $\sum_{i=1}^{n} C_{i,2} = \sum_{i=1}^{n} C_{i,1} + \sum_{i=1}^{n} p_{i,2}$, the schedule is optimal. Suppose set A is empty. At this moment, say at iteration ε, the profile of the current schedule is $(\varepsilon, \varepsilon+1, \varepsilon')$, with $\varepsilon' \geqslant \varepsilon + 2$ (see Fig. 3). Then, the remaining jobs of B are scheduled according to SPT/FAM algorithm. We denote by γ the number of unscheduled jobs at step ε, i.e. $\gamma = n - \varepsilon$. γ is either odd or even and we set $\gamma = 2k + h$ with $h \in \{0, 1\}$. We have

$$\sum_{i=1}^{n} C_i = \sum_{i=1}^{\varepsilon+h} \mathscr{C}_{[i]} + \sum_{i=\varepsilon+h+1}^{n} \mathscr{C}_{[i]}.$$

- Let us consider the first $\varepsilon + h$ jobs.

$$\sum_{i=1}^{\varepsilon+h} \mathscr{C}_{[i]} = \sum_{i=1}^{\varepsilon+h} \mathscr{C}_{[i],1} + \sum_{i=1}^{\varepsilon+h} p_{[i],2}.$$

The value $\sum_{i=1}^{\varepsilon+h} \mathscr{C}_{[i],1}$ cannot be reduced since the server is working without idle time in $[0, \varepsilon+h]$. Thus,

$$\sum_{i=1}^{\varepsilon+h} \mathscr{C}_{[i],1} = p_{[1],1} + p_{[2],1} + \cdots + p_{[\varepsilon+h],1} = \frac{(\varepsilon+h)(\varepsilon+h+1)}{2} = K$$

$$\Rightarrow \sum_{i=1}^{\varepsilon+h} \mathscr{C}_{[i]} = K + \sum_{i=1}^{\varepsilon+h} p_{[i],2}.$$

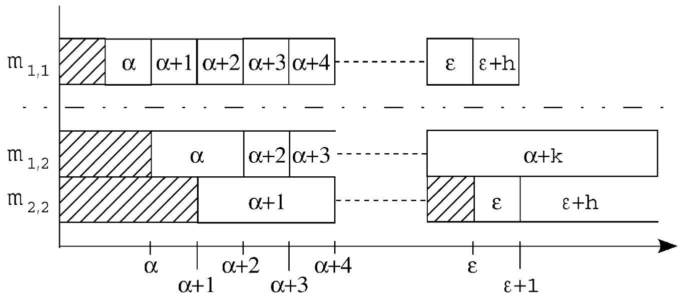

Fig. 3. Example of profile with $h = 1$.

The value $\sum_{i=1}^{\varepsilon+h} p_{[i],2}$ cannot be reduced since this sum contains the $\varepsilon+h$ smallest $p_{i,2}$, thus $\sum_{i=1}^{\varepsilon+h} \mathscr{C}_{[i]}$ is minimum.

• Let consider the second part of the schedule. Because jobs are scheduled according to SPT/FAM algorithm, they are assigned alternatively to M_1 and M_2. Thus, there is no idle time on M_1 and M_2 and we have:

$$\mathscr{C}_{[n]} + \mathscr{C}_{[n-1]} = T_1 + T_2 + \sum_{i=1}^{n} p_{[i],2},$$

$$\mathscr{C}_{[n]} = \mathscr{C}_{[n-2]} + p_{[n],2},$$

$$\mathscr{C}_{[n-1]} = \mathscr{C}_{[n-3]} + p_{[n-1],2}$$

$$\Rightarrow \mathscr{C}_{[n-2]} + \mathscr{C}_{[n-3]} = T_1 + T_2 + \sum_{i=1}^{n-2} p_{[i],2}$$

$$\Rightarrow \mathscr{C}_{[\ell]} + \mathscr{C}_{[\ell-1]} = T_1 + T_2 + \sum_{i=1}^{\ell} p_{[i],2}, \ \forall \ell, \ n-2k+2 \leqslant \ell \leqslant n. \tag{11}$$

Lemma 4 indicates that $T_1 + T_2 + \sum_{i=1}^{\ell} p_{[i],2}$ is a lower bound for $\mathscr{C}_{[\ell]} + \mathscr{C}_{[\ell-1]}, \forall \ell, 1 \leqslant \ell \leqslant n$, thus, we deduce that

$$\mathscr{C}_{[\varepsilon+h+1]} + \mathscr{C}_{[\varepsilon+h+2]} \geqslant T_1 + T_2 + \sum_{i=1}^{\varepsilon+h+1} p_{[i],2},$$

$$\vdots$$

$$\mathscr{C}_{[n]} + \mathscr{C}_{[n-1]} \geqslant T_1 + T_2 + \sum_{i=1}^{n} p_{[i],2}.$$

We set

$$LB = \frac{n-(\varepsilon+h)}{2} \times (T_1 + T_2) + \sum_{w=0}^{\frac{1}{2}(n-(\varepsilon+h+1))} \sum_{i=1}^{\varepsilon+h+2w+1} p_{[i],2}.$$

Because of Eq. (11), the schedule obtained by Algorithm 1 for jobs in position $\varepsilon+h+1$ to n is such that $\sum_{i=\varepsilon+h+1}^{n} \mathscr{C}_{[i]} = LB$, and thus is optimal.

For any schedule $\sigma, \sum_{i=1}^{n} C_i(\sigma) = \sum_{i=1}^{\varepsilon+h} C_i(\sigma) + \sum_{i=\varepsilon+h+1}^{n} C_i(\sigma)$. For the schedule σ^a obtained by Algorithm 1, $\sum_{i=1}^{n} C_i(\sigma^a) = \sum_{i=1}^{\varepsilon+h} C_i(\sigma^a) + \sum_{i=\varepsilon+h+1}^{n} C_i(\sigma^a)$. We have shown that $\forall \sigma, \sum_{i=1}^{\varepsilon+h} C_i(\sigma) \geqslant \sum_{i=1}^{\varepsilon+h} C_i(\sigma^a)$ and $\forall \sigma$, $\sum_{i=\varepsilon+h+1}^{n} C_i(\sigma) \geqslant \sum_{i=\varepsilon+h+1}^{n} C_i(\sigma^a)$. Thus, $\forall \sigma$, $\sum_{i=1}^{n} C_i(\sigma) \geqslant \sum_{i=1}^{n} C_i(\sigma^a)$ and so, the obtained value of $\sum_{i=1}^{n} C_i(\sigma^a)$ is optimum. $\square$

6. Extension of result

We show now that an optimal solution for the $FH2,(1,P2)|p_{i,1}=1,nowait|\sum C_i$ problem is also an optimal solution for the $FH2,(1,P2)|p_{i,1}=1|\sum C_i$ problem.

Theorem 6. *An optimal solution for the* $FH2,(1,P2)|nowait, p_{i,1}=1|\sum C_i$ *problem is also an optimal solution for the* $FH2,(1,P2)|p_{i,1}=1|\sum C_i$ *problem.*

Proof. We assume, without loss of generality, that jobs are numbered according to the sequence on the machine at the first stage. Consider the problem $FH2,(1,P2)|p_{i,1}=1|\sum C_i$. There always exist an optimal solution where all jobs are performed at the first stage without idle times. We have a positive idle time Δ_i between the completion time of $o_{i,1}$ and the starting time of $o_{i,2}$, for all $i=1,\ldots,n$. Thus $\Delta_i = C_{i,2} - C_{i,1} - p_{i,2}, \ \forall i=1,\ldots,n$.

Because we have no idle time at the first stage, we have

$$\sum_{i=1}^{n} C_{i,2} = \sum_{i=1}^{n} C_{i,1} + \sum_{i=1}^{n} p_{i,2} + \sum_{i=1}^{n} \Delta_i$$

$$\Rightarrow \sum_{i=1}^{n} C_{i,2} = \frac{n(n+1)}{2} + \sum_{i=1}^{n} p_{i,2} + \sum_{i=1}^{n} \Delta_i = K' + \sum_{i=1}^{n} \Delta_i.$$

Note that $\Delta_i = C_{i,2} - C_{i,1} - p_{i,2} = C_{i,2} - p_{i,2} - (C_{i-1,1}+1)$.

Let consider now the problem $FH2,(1,P2)|p_{i,1}=1,nowait|\sum C_i$. According to the no-wait constraint, we have no idle time between the first operation and the second operation of any job. However, we have a positive idle time δ_i between the completion time of operation $o_{i-1,1}$ and the starting time of operation $o_{i,1}$, for all $i=1,\ldots,n$. We denote by C_i' the completion time of job i for this problem. Thus $\delta_i = C_{i,1}' - C_{i-1,1}' - p_{i,1}, \ \forall i=1,\ldots,n$.

Thus

$$\sum_{i=1}^{n} C_{i,2}' = \sum_{i=1}^{n} C_{i,1}' + \sum_{i=1}^{n} p_{i,2},$$

we have

$$C'_{i,1} = \sum_{j=1}^{i} p_{j,1} + \sum_{j=1}^{i} \delta_j \quad \Rightarrow \quad \sum_{i=1}^{n} C'_{i,1} = \sum_{i=1}^{n}\sum_{j=1}^{i} p_{j,1} + \sum_{i=1}^{n}\sum_{j=1}^{i} \delta_j$$

$$\Rightarrow \sum_{i=1}^{n} C'_{i,1} = \sum_{i=1}^{n} (n-i+1)p_{i,1} + \sum_{i=1}^{n} (n-i+1)\delta_i = \frac{n(n+1)}{2} + \sum_{i=1}^{n} (n-i+1)\delta_i,$$

thus

$$\sum_{i=1}^{n} C'_{i,2} = \frac{n(n+1)}{2} + \sum_{i=1}^{n} p_{i,2} + \sum_{i=1}^{n} (n-i+1)\delta_i = K' + \sum_{i=1}^{n} (n-i+1)\delta_i.$$

Suppose that an optimal solution to the problem with the no-wait constraint is given by the sequence σ^* at the first stage and the vector of completion times $C'_{i,2}(\sigma^*)$ at the second stage. The sequence σ^* is optimal if and only if for all sequence $\sigma \neq \sigma^*$ we have $\sum_{i=1}^{n} C'_{i,2}(\sigma) \geqslant \sum_{i=1}^{n} C'_{i,2}(\sigma^*)$. That is to say if and only if $\sum_{i=1}^{n} (n-i+1)\delta_i(\sigma) \geqslant \sum_{i=1}^{n}(n-i+1)\delta_i(\sigma^*)$. We build a solution for the problem without the no-wait constraint as follows: we keep the same sequence at the first stage and we shift to the left the operations at the first stage if possible, so that there is no idle time between the first and the last operation. The completion times at the second stage are unchanged because of the resource constraints (otherwise the sequence σ^* would not be optimal for the problem with the no-wait constraint). We have to show that this solution is optimal for the problem without the no-wait constraint.

Let k be the first job of σ^* shifted to the left. We have $\Delta_k(\sigma^*) = \delta_k(\sigma^*)$.

Let $k+i$ be the $i+1$th job of σ^* shifted to the left. We have $\Delta_{k+i}(\sigma^*) = \sum_{j=0}^{i} \delta_{k+j}(\sigma^*)$. Thus

$$\sum_{i=1}^{n} C_{i,2}(\sigma^*) = K' + \sum_{i=1}^{n} \Delta_i(\sigma^*) = K' + \sum_{i=1}^{k-1} \Delta_i(\sigma^*) + \sum_{i=k}^{n} \Delta_i(\sigma^*)$$

$$\Rightarrow \sum_{i=1}^{n} C_{i,2}(\sigma^*) = K' + \delta_k(\sigma^*) + (\delta_k(\sigma^*) + \delta_{k+1}(\sigma^*)) + \cdots + (\delta_k(\sigma^*) + \cdots + \delta_n(\sigma^*))$$

with $\Delta_i(\sigma^*) = 0,\ \forall i,\ 1 \leqslant i \leqslant k-1$.

Thus

$$\sum_{i=l}^{n} C_{i,2}(\sigma^*) = K' + \sum_{i=k}^{n} (n-i+1)\delta_i(\sigma^*).$$

Because of the definition of σ^* for all sequence σ', we have

$$\sum_{i=1}^{n} (n-i+1)\delta_i(\sigma') \geqslant \sum_{i=1}^{n} (n-i+1)\delta_i(\sigma^*).$$

Because

$$\sum_{i=1}^{n} (n-i+1)\delta_i(\sigma^*) \geqslant \sum_{i=k}^{n} (n-i+1)\delta_i(\sigma^*)$$

for all sequence σ', $\sum_{i=1}^{n} C_{i,2}(\sigma') \geqslant \sum_{i=1}^{n} C_{i,2}(\sigma^*)$ and thus σ^* is an optimal sequence for the problem without the no-wait constraint. $\square$

Thus we deduce that Algorithm 1 gives also an optimal solution to the $FH2,(1,P2)|p_{i,1}=1|\sum C_i$ problem.

7. Conclusion

In this paper, we consider a parallel machine scheduling problem with a server in a computer system, modeled by a two-stage hybrid flow shop scheduling problem with a no-wait constraint. We propose a mathematical formulation for the problem and we propose two polynomial time algorithms for particular cases. We show that the two-stage hybrid flow shop problem is at least as difficult as the parallel machine scheduling problem with a single server. We deduce that the problem under consideration with real processing times is NP-hard. However, with integer processing times, we propose an $O(n\log(n))$ algorithm that solves the problem optimally. Finally, we show that an optimal solution to the two-stage hybrid flow shop problem with the no-wait constraint is an optimal solution to the problem without the no-wait constraint.

The further directions of this work are to consider the problem with m machines, and to consider the communication process between the operation on the server and the corresponding operation on the machine, with more details.

References

[1] Blazewicz J, Dell'Olmo P, Drozdowski M. Scheduling of client-server applications. International Transactions in Operational Research 1999;6:345–63.
[2] Koulamas CP. Scheduling two parallel semiautomatic machines to minimize machine interference. Computers & Operations Research 1996;23(10):945–56.
[3] Kravchenko SA, Werner F. Parallel machine scheduling problems with a single server. Mathematical and Computational Modelling 1997;26:1–11.
[4] Hall NG, Potts CN, Sriskandarajah C. Parallel machine scheduling with a common server. Discrete Applied Mathematics 2000;102:223–43.
[5] Brucker P, Dhaenens-Flipo C, Knust S, Kravchenko SA, Werner F. Complexity results for a parallel machine problems with a single server. Osnabrück Schriften zur Mathematik 2000;219:1–29.
[6] Glass CA, Shafransky YM, Strusevich VA. Scheduling for parallel dedicated machines with a single server. Naval Research Logistics 2000;47:304–28.
[7] Kravchenko SA, Werner F. A heuristic algorithm for minimizing mean flow time with unit setups. Information Processing Letters 2001;79:291–6.
[8] Gupta JND. Two-stage, hybrid flowshop scheduling problem. Operational Research Society 1988;39(4):359–64.
[9] Hoogeveen JA, Lenstra JK, Veltman B. Preemptive scheduling in a two-stage multiprocessor flowshop is np-hard. European Journal of Operational Research 1996;86:172–5.
[10] Vignier A, Billaut J-C, Proust C. Les flowshop hybrides: état de l'art (in french). RAIRO-RO 1999;2(33):117–83.
[11] Linn R, Zhang W. Hybrid flowshop scheduling: a survey. Computers and Industrial Engineering 1999;(37):57–61.
[12] Pinedo M. Scheduling: Theory, algorithms and systems. Englewood Cliffs, NJ: Prentice-Hall; 1995.
[13] Graham RL, Lawler EL, Lenstra JK, Kan AHGR. Optimization and approximation in deterministic sequencing and scheduling theory: a survey. Annals of Discrete Mathematics 1979;(5):287–326.
[14] Kaspi M, Montreuil B. On the scheduling of identical parallel processes with arbitrary initial processor available time. Research Report 88-12, School of Industrial Engineering, Purdue University; 1988.