Journal of the Operational Research Society (2002) 53, 1239–1246 0160-5682/02 $15.00
www.palgrave-journals.com/jors

A dynamic programming algorithm for scheduling jobs in a two-machine open shop with an availability constraint

T Lorigeon, J-C Billaut* and J-L Bouquard
Université de Tours, Tours, France

This paper studies a two-machine open shop scheduling problem with an availability constraint, ie we assume that a machine is not always available and that the processing of the interrupted job can be resumed when the machine becomes available again. We consider the makespan minimization as criterion. This problem is NP-hard. We develop a pseudo-polynomial time dynamic programming algorithm to solve the problem optimally when the machine is not available at time $s > 0$. Then, we propose a mixed integer linear programming formulation, that allows to solve instances with up to 500 jobs optimally in less than 5 min with CPLEX solver. Finally, we show that any heuristic algorithm has a worst-case error bound of 1.
Journal of the Operational Research Society (2002) **53,** 1239–1246. doi:10.1057/palgrave.jors.2601421

Keywords: scheduling; open shop; dynamic programming; integer programming

Introduction

Classical scheduling problems assume that the machines are always available for processing jobs. This assumption may not be true if one assumes that some preventive maintenance operations are already scheduled, or if resources are humans who only work in some periods, known in advance.

We consider a scheduling problem with two machines, where each job is composed of two operations. One operation has to be scheduled on the first machine and the other operation on the second machine. We assume that the processing route is not fixed, ie the operation on the first machine can be scheduled before or after the operation on the second machine. The problem under consideration is called a 'two-machine' open shop scheduling problem, and is denoted by $O2$. An open shop problem can model some industrial processes and can be encountered in testing facilities for instance, where items have to go through a series of tests, in a non-specified order. More precisely, scheduling the tests of components of an electronic system can be considered as an open shop scheduling problem. In the two-machine open shop scheduling problem we consider, there is an availability constraint, on the first machine. We assume that if a job cannot be finished before the unavailable period, then it can continue after the machine is available again (called the '*resumable*' case). We search for a schedule that minimizes the makespan, denoted by $C_{\max}$, and equal to the maximum completion time of jobs. Several notations have been proposed in the literature for these problems. According to the standard three-field notation of scheduling problems and to the notation of Kubiak *et al*,[1] the problem is denoted by $O2, h_{11}|rs|C_{\max}$, where h_{kj} indicates that k unavailable periods occur on the machine j and rs stands for '*resumable activities*'.[2]

In the literature, most of the deterministic scheduling problems where machines are not continuously available for processing, deal with one-machine problems or parallel machine problems.[3] Some papers deal with the flow shop problem, ie the scheduling problem for which the operation on the first machine has to be scheduled before the operation on the second machine, etc. When there are only two machines, the problem is denoted by $F2$. Lee[4] shows that the $F2, h_{11}|rs|C_{\max}$ problem is NP-hard in the ordinary sense and proposes a pseudo-polynomial dynamic programming algorithm and a heuristic algorithm, with a worst-case error bound of 1/2. The author extends the hypotheses[5] and proposes new resolution methods for new versions of this problem. Cheng and Wang[6] propose a heuristic algorithm for the $F2, h_{11}|rs|C_{\max}$ problem with a worst-case error bound of 1/3. The authors also consider the case with an availability constraint on each machine.[7] For the $F2, h_{kj}|rs|C_{\max}$ problem, Kubiak *et al*[1] propose complexity results, properties and a branch-and-bound algorithm and Blazewicz *et al*[8] propose several heuristic algorithms. Sanlaville and Schmidt[9] and Schmidt[10] present a review of scheduling problems if machines are not always available. Blazewicz and Formanowicz[11] consider the two-machine

*Correspondence: J-C Billaut, Laboratoire d'Informatique, Université de Tours, Ecole d'Ingénieurs en Informatique pour l'Industrie, 64 avenue Jean Portalis, 37200 Tours, France.
E-mail: billaut@e3i.univ-tours.fr

open shop problem. The authors consider non-preemptable, resumable and preemptable cases and propose complexity results and mathematical programming formulations. Breit *et al*[12] show that the problem considered in this paper is NP-hard and propose an approximation algorithm with a worst-case ration of 4/3.

First, we present the notations of the problem. Then, we propose a pseudo-polynomial dynamic programming algorithm to solve this problem and a mixed integer linear programming (MILP) formulation of the problem. We show that any heuristic algorithm has a worst-case error bound of 1. Finally, computational experiences are proposed and show that the integer linear program allows to solve optimally instances with up to 500 jobs in less than 5 min.

Notations

We note $S = \{J_1, \ldots, J_n\}$ the set of jobs to schedule, O_{ij} the operation j of job J_i, $1 \leqslant i \leqslant n$, $j \in \{1, 2\}$, M_1 and M_2 machine 1 and machine 2, a_i and b_i the processing times for J_i on M_1 and M_2, respectively, and we say that M_1 is unavailable for processing jobs from s to t. We assume that $s > 0$, otherwise the problem is polynomially solvable in $O(n)$ time.[13] Without loss of generality we assume that only M_1 is unavailable for job processing during period $(t - s)$. When the routing of jobs is decided, we denote by X the set of jobs that are processed first on M_1 and then on M_2, and by Y the set jobs that are processed first on M_2 and then on M_1.

First, we recall the solution of Gonzales and Sahni[14] for the $O2||C_{\max}$ problem. Let us define:

$$U = \{J_i/a_i \leqslant b_i, 1 \leqslant i \leqslant n\}, \quad V = \{J_i/a_i > b_i, 1 \leqslant i \leqslant n\}$$

We denote by J_k the job such that $\min(a_k, b_k) = \max_{i=1}^{n} \min(a_i, b_i)$. If $a_k = \min(a_k, b_k)$, then an optimal schedule is represented in Figure 1a. If $b_k = \min(a_k, b_k)$, then an optimal schedule is represented in Figure 1b. In both cases, the optimal makespan value, denoted by $C^*_{\max}(O2)$, is easy to determine:

$$C^*_{\max}(O2) = \max\left(\sum_{i=1}^{n} a_i, \sum_{i=1}^{n} b_i, \max_{i=1}^{n}(a_i + b_i)\right)$$

Notice that in both cases, a symetric solution can be proposed by the reverse sequences on each machine.

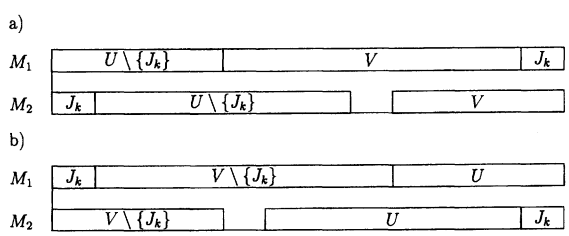

Figure 1 Optimal solution for the $O2||C_{\max}$ problem.

General case study

We suppose that $0 < s < \sum_{i=1}^{n} a_i$ and $t < \sum_{i=1}^{n} b_i$, otherwise there is no problem. This problem has been proved to be NP-hard.[2,11,12]

Firstly, we propose a dominant solution for the general problem. Secondly, we propose an optimal pseudo-polynomial time dynamic programming algorithm. Thirdly, we propose a MILP formulation of the problem, that can be used to solve instances with an important number of jobs. Finally, we show that any heuristic algorithm has a worst-case error bound of 1.

Dominant solution

Let consider a solution S. We denote by $t_{i,j}$ the starting time of operation O_{ij}, and $a//b$ stands for the concatenation of a and b.

Proposition 1 *To the solution S with the unavailable period (s, t) and an interrupted job J_i, it corresponds a solution S' with the same sequences as S, an unavailable period (s', t') and no interrupted job, such that $t' - s' = t - s$ and $C_{\max}(S) = C_{\max}(S')$. S' is called 'the equivalent solution, without interruption'.*

Proof If $J_i \in X$, we consider that $s' = t_{i,1}$ and $t' = s' + (t - s)$ with $t_{i,1}$ the starting time of J_i on M_1. If $J_i \in Y$, we consider that $t' = t_{i,1} + a_i + (t - s)$ and $s' = t' - (t - s)$. These solutions are equivalent to S in terms of makespan. □

Let us define:

$$XP = \{J_i \in X/t_{i,1} + a_i \leqslant s\}, \quad XS = \{J_i \in X/t_{i,1} + a_i > s\},$$
$$YP = \{J_i \in Y/t_{i,2} < s\}, \quad YS = \{J_i \in Y/t_{i,2} \geqslant s\}.$$

Proposition 2 *There exists an optimal solution such that on M_1, the sequence is defined by $XP//YP//XS//YS$ and on M_2 the sequence is defined by $YP//YS/XP//XS$, and the order of the jobs in XP, XS, YP and YS is the same on M_1 and M_2* (see Figure 2).

Proof Consider an optimal solution S. We build a solution S' as follows. Consider two consecutive jobs $J_{y_p} \in YP$ and $J_{x_p} \in XP$ in this order on M_1. Then, it is possible to permute J_{x_p} and J_{y_p} on M_1 without increasing the makespan value of S. In the same way, if we consider two consecutive jobs $J_{y_s} \in YS$ and $J_{x_s} \in XS$ in this order on M_1, it is possible to permute J_{x_s} and J_{y_s} on M_1 in S' without increasing the makespan value of S. So, the sequence of jobs on M_1 is $XP//YP//XS//YS$. For similar reasons on M_2, it is possible

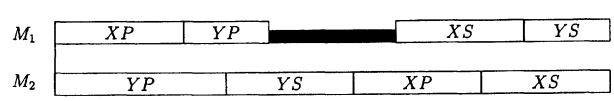

Figure 2 Dominant solution for $O2$, $h_{11}|rs|C_{\max}$.

to sequence the jobs without increasing the makespan value as follows: $YP//YS//XP//XS$. Because S is optimal, S' is optimal too. □

One can note that in Figure 2, the sets are not interrupted, as if it was possible to shift the unavailable period to the right or to the left. In fact the equivalent solutions without interruption are considered here (see Proposition 1).

Proposition 3 *At least, one of the two sets YP and XS is empty.*

Proof Suppose that $YP \neq \emptyset$ and $XS \neq \emptyset$. Consider J_p the last job of YP and J_s the first job of XS. Without increasing the makespan value, it is possible to permute operations O_{p2} and O_{s1} and to add J_s to XP and J_p to YS. This process can be iterated until one of the two sets becomes empty. □

Dynamic programming for O2, $h_{11}|rs|C_{\max}$

The jobs are re-indexed following Johnson's rule.[15] We define the state of the system by the vector $\sigma = (k, u_1, v_1, w_1, x_1, u_2, v_2, w_2, x_2, y_2)$ (see Figure 3 if $k = 0$ and Figure 4 if $k = 1$), with:

- k equal to 0 (or 1) if YP (XS respectively) is empty (see Proposition 3),
- u_1 completion time of the last job of XP, on M_1,
- v_1 starting time of the first job of XS if $k = 0$, or YP if $k = 1$, on M_1,
- w_1 completion time of the last job of XS if $k = 0$, or YP if $k = 1$, on M_1,
- x_1 earliest starting time of the first job of YS, on M_1,
- u_2 starting time of the first job of YS if $k = 0$ of YP if $k = 1$, on M_2,
- v_2 completion time of the last job of YS if $k = 0$ or YP if $k = 1$, on M_2,
- w_2 starting time of the first job of XP if $k = 0$ or YS if $k = 1$, on M_2,
- x_2 completion time of the last job of XP if $k = 0$ or YS if $k = 1$, on M_2,
- y_2 earliest starting time of the first job of XS if $k = 0$ or XP if $k = 1$, on M_2.

The variation of the indices is the following. If $k = 0$, $x_1 \in [t, GS+(t-s)]$, $w_1 \in [t, x_1]$, $v_1 \in [\max(s-a_{\max}+1, 0), s]$, $u_1 \in [0, v_1]$, $y_2 \in [t+1, GS+(t-s)]$, $x_2 \in [u_1, y_2]$, $w_2 \in [0, x_2]$, $v_2 \in [0, w_2]$, and $u_2 \in [0, \min(v_2, x_1)]$. If $k = 1, x_1 \in [t, GS+(t-s)]$, $w_1 \in [t, \min(t+a_{\max}-1, x_1]$, $v_1 \in [0, s]$, $u_1 \in [0, v_1]$, $y_2 \in [0, GS+(t-s)]$, $x_2 \in [0, y_2]$, $w_2 \in [0, x_2]$, $v_2 \in [0, \min(w_2, s-1)]$ and $u_2 \in [0, \min(v_2, v_1)]$.

We set $A_i = \sum_{j=1}^{i} a_j$, $B_i = \sum_{j=1}^{i} b_j$ and $GS = C^*_{\max}(O2) = \max\left(\sum_{i=1}^{n} a_i, \sum_{i=1}^{n} b_i, \max_{i=1}^{n}(a_i + b_i)\right)$. The function $(f_{i,1}(\sigma_i), f_{i,2}(\sigma_i))$ corresponds to the completion time of the jobs of $\{J_1, \ldots, J_i\}$ on M_1 and M_2, respectively, if the system state is σ_i. This function could be considered like a two-dimensional vector, as if the problem involved two criteria. The maximum completion time is the maximum between $f_{i,1}(\sigma_i)$ and $f_{i,2}(\sigma_i)$. The jobs are ordered following Johnson's rule and considered one after the other and the decision variable is to decide in which set belongs job J_i. More precisely, in XS or XP the job is included at the end of the sequence to obtain Johnson's order, and in YS or YP the job is included at the beginning of the sequence to obtain Johnson's reverse order. With $\sigma = (k, u_1, v_1, w_1, x_1, u_2, v_2, w_2, x_2, y_2)$, the initial conditions are the following:

$$f_{1,1}(\sigma) = \begin{cases} x_1 & \text{if } k=0, u_1=a_1, u_1 \leqslant w_2, w_1-v_1-(t-s)=0, v_2=u_2, x_2-w_2=b_1 \\ x_1 & \text{if } k=0, u_1=0, w_1-v_1-(t-s)=a_1, ((v_1<s, w_1>t) \text{ or } (v_1=s)), v_2=u_2, x_2=w_2 \\ \max(x_1, v_2)+a_1 & \text{if } k=0, u_1=0, w_1-v_1-(t-s)=0, v_2-u_2=b_1, x_2=w_2 \\ x_1 & \text{if } k=1, u_1=a_1, w_1-v_1-(t-s)=0, v_2=u_2, x_2=w_2 \\ x_1 & \text{if } k=1, u_1=0, w_1-v_1-(t-s)=a_1, ((v_1<s, w_1>t) \text{ or } (w_1=t)), v_2 \leqslant v_1, v_2-u_2=b_1, x_2=w_2 \\ \max(x_2, x_1)+a_1 & \text{if } k=1, u_1=0, w_1-v_1-(t-s)=0, v_2=u_2, x_2-w_2=b_1 \\ \infty & \text{otherwise} \end{cases}$$

$$f_{1,2}(\sigma) = \begin{cases} y_2 & \text{if } k=0, u_1=a_1, u_1 \leqslant w_2, w_1-v_1-(t-s)=0, v_2=u_2, x_2-w_2=b_1 \\ \max(w_1, y_2)+b_1 & \text{if } k=0, u_1=0, w_1-v_1-(t-s)=a_1, ((v_1<s, w_1>t) \text{ or } (v_1=s)), v_2=u_2, x_2=w_2 \\ y_2 & \text{if } k=0, u_1=0, w_1-v_1-(t-s)=0, v_2-u_2=b_1, x_2=w_2 \\ \max(u_1, y_2)+b_1 & \text{if } k=1, u_1=a_1, w_1-v_1-(t-s)=0, v_2=u_2, x_2=w_2 \\ y_2 & \text{if } k=1, u_1=0, w_1-v_1-(t-s)=a_1, ((v_1<s, w_1>t) \text{ or } (w_1=t)), v_2 \leqslant v_1, v_2-u_2=b_1, x_2=w_2 \\ y_2 & \text{if } k=1, u_1=0, w_1-v_1-(t-s)=0, v_2=u_2, x_2-w_2=b_1 \\ \infty & \text{otherwise} \end{cases}$$

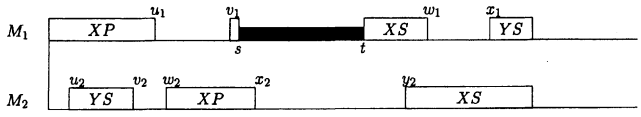

Figure 3 State of the system for dynamic programming, if $k = 0$, eg $YP = \emptyset$.

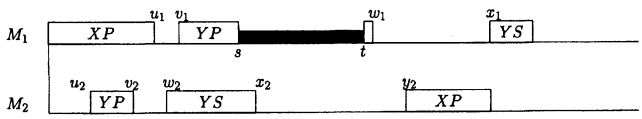

Figure 4 State of the system for dynamic programming, if $k = 1$, eg $XS = \emptyset$.

We set $\sigma = (k, u_1, v_1, w_1, x_1, u_2, v_2, w_2, x_2, y_2)$, and we distinguish four cases:

1. If $k=0, \max(u_1, w_2)+b_i \leqslant x_2, u_1 \geqslant a_i, u_1+w_1-v_1-(t-s) \leqslant A_i$ and $v_2-u_2+x_2-w_2 \leqslant B_i, ((v_1 < s, w_1 > t)$ or $(v_1 = s))$, then J_i can be added to XP and we have:

$$f_{i,1}^{XP}(\sigma) = f_{i-1,1}(k, u_1 - a_i, v_1, w_1, x_1, u_2, v_2, w_2, x_2 - b_i, y_2)$$
$$f_{i,2}^{XP}(\sigma) = f_{i-1,2}(k, u_1 - a_i, v_1, w_1, x_1, u_2, v_2, w_2, x_2 - b_i, y_2)$$

else if $k=1, u_1 \geqslant a_i, u_1+w_1-v_1-(t-s) \leqslant A_i$ and $v_2-u_2+x_2-w_2+b_i \leqslant B_i, ((v_1 < s, w_1 > t)$ or $(w_1 = t))$, then J_i can be added to XP and we have:

$$f_{i,1}^{XP}(\sigma) = f_{i-1,1}(k, u_1 - a_i, v_1, w_1, x_1, u_2, v_2, w_2, x_2, y_2)$$
$$f_{i,2}^{XP}(\sigma) = \max(f_{i-1,2}(k, u_1 - a_i, v_1, w_1, x_1, u_2, v_2, w_2, x_2, y_2),\ u_1) + b_i$$

else

$$f_{i,1}^{XP}(\sigma) = \infty, \quad f_{i,2}^{XP}(\sigma) = \infty$$

2. If $k=0, w_1-a_i \leqslant t, w_1-v_1-(t-s)=a_i, u_1+w_1-v_1-(t-s) \leqslant A_i$ and $v_2-u_2+x_2-w_2+b_i \leqslant B_i, ((v_1 < s, w_1 > t)$ or $(v_1 = s))$, J_i can be added to XS and we have:

$$f_{i,1}^{XS}(\sigma) = f_{i-1,1}(k, u_1, s, t, x_1, u_2, v_2, w_2, x_2, y_2)$$
$$f_{i,2}^{XS}(\sigma) = \max(f_{i-1,2}(k, u_1, s, t, x_1, u_2, v_2, w_2, x_2, y_2),\ w_1) + b_i$$

else if $k=0,\ w_1 - a_i > t, u_1 + w_1 - v_1 - (t-s) \leqslant A_i$ and $v_2-u_2+x_2-w_2+b_i \leqslant B_i, ((v_1 < s, w_1 > t)$ or $(v_1 = s))$, J_i can be added to XS we have:

$$f_{i,1}^{XS}(\sigma) = f_{i-1,1}(k, u_1, v_1, w_1 - a_i, x_1, u_2, v_2, w_2, x_2, y_2)$$
$$f_{i,2}^{XS}(\sigma) = \max(f_{i-1,2}(k, u_1, v_1, w_1 - a_i, x_1, u_2, v_2, w_2, x_2, y_2),\ w_1) + b_i$$

else

$$f_{i,1}^{XS}(\sigma) = \infty, \quad f_{i,2}^{XS}(\sigma) = \infty$$

3. If $k = 1, v_1 + a_i < s, v_2 - u_2 \geqslant b_i, v_1 \geqslant u_2 + b_i, u_1 + w_1 - v_1 - (t-s) \leqslant A_i$ and $v_2 - u_2 + x_2 - w_2 \leqslant B_i, ((v_1 < s, w_1 > t)$ *or* $(w_1 = t))$, J_i can be added to YP and we have:

$$f_{i,1}^{YP}(\sigma) = f_{i-1,1}(k, u_1, v_1 + a_i, w_1, x_1, u_2 + b_i, v_2, w_2, x_2, y_2)$$
$$f_{i,2}^{YP}(\sigma) = f_{i-1,2}(k, u_1, v_1 + a_i, w_1, x_1, u_2 + b_i, v_2, w_2, x_2, y_2)$$

else if $k = 1, v_1 + a_i \geqslant s, w_1 - v_1 - (t-s) = a_i, v_2 - u_2 \geqslant b_i, v_1 \geqslant u_2 + b_i, u_1 + w_1 - v_1 - (t-s) \leqslant A_i$ and $v_2 - u_2 + x_2 - w_2 \leqslant B_i, ((v_1 < s, w_1 > t)$ *or* $(w_1 = t))$, J_i can be added to YP and we have:

$$f_{i,1}^{YP}(\sigma) = f_{i-1,1}(k, u_1, s, t, x_1, u_2, v_2, w_2, x_2, y_2)$$
$$f_{i,2}^{YP}(\sigma) = f_{i-1,2}(k, u_1, s, t, x_1, u_2, v_2, w_2, x_2, y_2)$$

else

$$f_{i,1}^{YP}(\sigma) = \infty, \quad f_{i,2}^{YP}(\sigma) = \infty$$

4. If $k = 0, x_1 + a_i \leqslant GS + (t-s), v_2 - u_2 \geqslant b_i, u_1 + w_1 - v_1 - (t-s) + a_i \leqslant A_i$ and $v_2 - u_2 + x_2 - w_2 \leqslant B_i, ((v_1 < s, w_1 > t)$ *or* $(v_1 = s))$, J_i can be added to YS and we have:

$$f_{i,1}^{YS}(\sigma) = \max(\max(x_1, u_2 + b_i) + A_i - (u_1 + w_1 - v_1 - (t-s)), f_{i-1,1}(k, u_1, v_1, w_1, x_1 + a_i, u_2 + b_i, v_2, w_2, x_2, y_2))$$
$$f_{i,2}^{YS}(\sigma) = f_{i-1,2}(k, u_1, v_1, w_1, x_1 + a_i, u_2 + b_i, v_2, w_2, x_2, y_2)$$

else if $k = 1, x_1 + a_i \leqslant GS + (t-s), x_2 - w_2 \geqslant b_i, u_1 + w_1 - v_1 - (t-s) + a_i \leqslant A_i$ and $v_2 - u_2 + x_2 - w_2 \leqslant B_i, ((v_1 < s, w_1 > t)$ *or* $(w_1 = t))$, J_i can be added to YS and we have:

$$f_{i,1}^{YS}(\sigma) = \max(\max(w_2 + b_i, x_1) + A_i - (u_1 + w_1 - v_1 - (t-s)), f_{i-1,1}(k, u_1, v_1, w_1, x_1 + a_i, u_2, v_2, w_2 + b_i, x_2, y_2))$$
$$f_{i,2}^{YS}(\sigma) = f_{i-1,2}(k, u_1, v_1, w_1, x_1 + a_i, u_2, v_2, w_2 + b_i, x_2, y_2)$$

else

$$f_{i,1}^{YS}(\sigma) = \infty, \quad f_{i,2}^{YS}(\sigma) = \infty$$

Finally, we set:

$$\max(f_{i,1}(\sigma),\ f_{i,2}(\sigma)) = \min_{\delta \in \{XP, XS, YP, YS\}} (\max(f_{i,1}^{\delta}(\sigma),\ f_{i,2}^{\delta}(\sigma)))$$

We want to minimize the makespan, eg the smallest value f_n with:

$$f_n = \min_{\sigma} \max(f_{n,1}(\sigma),\ f_{n,2}(\sigma))$$

The complexity of this algorithm is in $O(n \cdot s^2 \cdot GSW^4 \cdot GSS^3)$ with $GSW = GS + (t - s)$ and $GSS = GS - s$.

Mixed integer linear programming model

We consider that the jobs are numbered according to Johnson's rule. We build an optimal solution by considering the equivalent solution without interruption (Proposition 1) and the dominant solution (Proposition 2). Let us define the following decision variables: $\forall i,\ 1 \leqslant i \leqslant n$

- $XP_i = 1$ if $J_i \in XP$, 0 otherwise,
- $XS_i = 1$ if $J_i \in XS$, 0 otherwise,
- $YP_i = 1$ if $J_i \in YP$, 0 otherwise,
- $YS_i = 1$ if $J_i \in YS$, 0 otherwise.

The objective function is to minimize $Z = C_{\max}$. The constraints of the MILP are the following.

A job J_i belongs to only one set: $\forall i, 1 \leqslant i \leqslant n : XP_i + YP_i + XS_i + YS_i = 1$.

The unavailability period starts at time s and is finished at time t. Because YP or XS is empty (see Proposition 3), we have:

$$s \geqslant \sum_{i=1}^{n} a_i XP_i, \quad t \leqslant C_{\max} - \sum_{i=1}^{n} a_i YS_i$$

The makespan of the sequence is given by one of the six following paths: $(XP^{M_1}, XP^{M_2}, XS^{M_2})$ (Equation (1)), $(XP^{M_1}, YP^{M_1}, XS^{M_1}, XS^{M_2})$ (Equation (2)), $(YP^{M_2}, YP^{M_1}, XS^{M_1}, YS^{M_1})$ (Equation (3)), $(YP^{M_2}, YS^{M_2}, YS^{M_1})$ (Equation (4)), $(XP^{M_1}, YP^{M_1}, XS^{M_1}, YS^{M_1})$ (Equation (5)), and $(YP^{M_2}, YS^{M_2}, XP^{M_2}, XS^{M_2})$ (Equation (6)).

$$C_{\max} \geqslant \sum_{i=1}^{k} a_i XP_i + \sum_{i=k}^{n} b_i XP_i + \sum_{i=1}^{n} b_i XS_i, \quad \forall k, 1 \leqslant k \leqslant n \tag{1}$$

$$C_{\max} \geqslant \sum_{i=1}^{n} a_i XP_i + \sum_{i=1}^{n} a_i YP_i + (t - s) + \sum_{i=1}^{k} a_i XS_i + \sum_{i=k}^{n} b_i XS_i, \quad \forall k, 1 \leqslant k \leqslant n \tag{2}$$

$$C_{\max} \geqslant \sum_{i=1}^{k} b_i YP_i + \sum_{i=k}^{n} a_i YP_i + (t - s) + \sum_{i=1}^{n} a_i XS_i + \sum_{i=1}^{n} a_i YS_i, \quad \forall k, 1 \leqslant k \leqslant n \tag{3}$$

$$C_{\max} \geqslant \sum_{i=1}^{n} b_i YP_i + \sum_{i=1}^{k} b_i YS_i + \sum_{i=k}^{n} a_i YS_i, \quad \forall k,\ 1 \leqslant k \leqslant n \tag{4}$$

$$C_{\max} \geqslant (t - s) + \sum_{i=1}^{n} a_i \tag{5}$$

$$C_{\max} \geqslant \sum_{i=1}^{n} b_i \tag{6}$$

We can note, that even if the jobs of YP and YS have to be sequenced following Johnson's reverse order, because they are processed first on M_2 and then on M_1, the expression of the optimal makespan value of these subsequences is obtained by sequencing jobs according to Johnson's order, and by computing for all k, $1 \leqslant k \leqslant n$ the length of the path which is equal to $\sum_{i=1}^{k} a_i + \sum_{i=k}^{n} b_i$. This model contains $4n$ binary variables, 1 integer variable and $5n + 4$ constraints.

Heuristic

We denote by C_H the makespan obtained by heuristic H and by C^* the optimal makespan.

Proposition 4 *For any heuristic H,* $(C_H - C^*)/C^* \leqslant 1$.

Proof The worst possible makespan for the $O2,\ h_{11}||C_{\max}$ problem is equal to $\sum_{i=1}^{n} a_i + (t - s) + \sum_{i=1}^{n} b_i$. Two solutions can give this result: the first operations of the jobs are sequenced on M_1 and then on M_2, or the contrary.

A lower bound to the optimal makespan is equal to $\mathrm{LB}(O2, h_{11})$, where:

$$\mathrm{LB}(O2, h_{11}) = \max\left(\sum_{i=1}^{n} a_i + (t - s), \sum_{i=1}^{n} b_i, \max_{i=1}^{n}(a_i + b_i)\right)$$

We have $C_H \geqslant C^* \geqslant \mathrm{LB}(O2, h_{11})$.

It is clear that if we show that $(C_H - \mathrm{LB}(O2, h_{11}))/\mathrm{LB}(O2, h_{11}) \leqslant 1$, then we deduce easily that $(C_H - C^*)/C^* \leqslant 1$. We consider three cases.

1. $\mathrm{LB}(O2, h_{11}) = \sum_{i=1}^{n} a_i + (t - s)$

$$\frac{C_H - LB(O2, h_{11})}{LB(O2, h_{11})} = \frac{\sum_{i=1}^{n}(a_i + b_i) + (t - s) - \sum_{i=1}^{n} a_i - (t - s)}{\sum_{i=1}^{n} a_i + (t - s)} = \frac{\sum_{i=1}^{n} b_i}{\sum_{i=1}^{n} a_i + (t - s)}$$

Because

$$\sum_{i=1}^{n} a_i + (t - s) \geqslant \sum_{i=1}^{n} b_i, \frac{C_H - \mathrm{LB}(O2, h_{11})}{\mathrm{LB}(O2, h_{11})} \leqslant 1.$$

2. $\mathrm{LB}(O2, h_{11}) = \sum_{i=1}^{n} b_i$

$$\frac{C_H - \mathrm{LB}(O2, h_{11})}{\mathrm{LB}(O2, h_{11})} = \frac{\sum_{i=1}^{n}(a_i + b_i) + (t - s) - \sum_{i=1}^{n} b_i}{\sum_{i=1}^{n} b_i} = \frac{\sum_{i=1}^{n} a_i + (t - s)}{\sum_{i=1}^{n} b_i}$$

Because

$$\sum_{i=1}^{n} b_i \geqslant \sum_{i=1}^{n} a_i + (t-s), \frac{C_H - \mathrm{LB}(O2, h_{11})}{\mathrm{LB}(O2, h_{11})} \leqslant 1.$$

3. $\mathrm{LB}(O2, h_{11}) = \max_{i=1}^{n}(a_i + b_i)$

$$\frac{C_H - \mathrm{LB}(O2, h_{11})}{\mathrm{LB}(O2, h_{11})} = \frac{\sum_{i=1}^{n}(a_i + b_i) + (t-s) - \max_{i=1}^{n}(a_i + b_i)}{\max_{i=1}^{n}(a_i + b_i)}$$

$$\max_{i=1}^{n}(a_i + b_i) \geqslant \sum_{i=1}^{n} a_i + (t-s) \Rightarrow \sum_{i=1}^{n} a_i + (t-s) - \max_{i=1}^{n}(a_i + b_i) \leqslant 0.$$

Because

$$\max_{i=1}^{n}(a_i + b_i) \geqslant \sum_{i=1}^{n} b_i,$$

$$\max_{i=1}^{n}(a_i + b_i) \geqslant \sum_{i=1}^{n} b_i + \sum_{i=1}^{n} a_i + (t-s) - \max_{i=1}^{n}(a_i + b_i).$$

$$\Rightarrow \frac{C_H - \mathrm{LB}(O2, h_{11})}{\mathrm{LB}(O2, h_{11})} \leqslant 1 \qquad \square$$

Computational results

The MILP has been tested using CPLEX solver on a Pentium II 400 MHz with 128 Mb RAM. Processing times on M_1 are randomly generated in $\{0, \ldots, 100\}$. We distinguish nine cases for the generation of dates s and t, summarized in Table 1. In cases 1–3, the unavailability period starts in the first third on M_1, in cases 4–6 it starts in the second third and in cases 7–9 in the last third.

Table 1 Generation of dates s and t

Case 1: $s \in [0, \frac{1}{3}\sum a_i]$ and $t - s \in [0, \frac{1}{4}\sum a_i]$
Case 2: $s \in [0, \frac{1}{3}\sum a_i]$ and $t - s \in [\frac{1}{4}\sum a_i, \frac{1}{2}\sum a_i]$
Case 3: $s \in [0, \frac{1}{3}\sum a_i]$ and $t - s \in [\frac{1}{2}\sum a_i, \frac{3}{4}\sum a_i]$
Case 4: $s \in [\frac{1}{3}\sum a_i, \frac{2}{3}\sum a_i]$ and $t - s \in [0, \frac{1}{4}\sum a_i]$
Case 5: $s \in [\frac{1}{3}\sum a_i, \frac{2}{3}\sum a_i]$ and $t - s \in [\frac{1}{4}\sum a_i, \frac{1}{2}\sum a_i]$
Case 6: $s \in [\frac{1}{3}\sum a_i, \frac{2}{3}\sum a_i]$ and $t - s \in [\frac{1}{2}\sum a_i, \frac{3}{4}\sum a_i]$
Case 7: $s \in [\frac{2}{3}\sum a_i, \sum a_i]$ and $t - s \in [0, \frac{1}{4}\sum a_i]$
Case 8: $s \in [\frac{2}{3}\sum a_i, \sum a_i]$ and $t - s \in [\frac{1}{4}\sum a_i, \frac{1}{2}\sum a_i]$
Case 9: $s \in [\frac{2}{3}\sum a_i, \sum a_i]$ and $t - s \in [\frac{1}{2}\sum a_i, \frac{3}{4}\sum a_i]$

Then, the duration of the considered unavailability period is either small, medium or large. Processing times on M_2 are generated such that

$$\sum_{i=1}^{n} a_i + (t-s) \approx \sum_{i=1}^{n} b_i.$$

To do that, processing times on M_2 are randomly generated in

$$\left\{0, \ldots, \frac{2}{n}\left(\sum_{i=1}^{n} a_i + (t-s)\right)\right\}.$$

The number of jobs is $n \in \{200, 300, 400, 500\}$. For each value of n, 30 instances are generated. We present in Table 2, for each case and for each value of n, the mean computation time ($\bar{t}$), the minimum computation time ($t_{\min}$) and the maximum computation time ($t_{\max}$) in seconds of CPLEX solver.

These computational experiments show that the problem is in practice not difficult to solve optimally. Indeed, with $n = 500$ jobs, the maximum computation time over all the instances is less than 5 min. In fact, the real problem for computational experiments was to find difficult instances. So, other tests have been made with instances generated as indicated in the complexity proof,[12] in order to solve a PARTITION-like problem. The number of jobs is $n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 125, 150, 175, 200\}$. Processing times on M_1 have been randomly generated in $\{1, \ldots, 11\}$. Computational results are summarized in Table 3. Because the optimal makespan is equal to $(1/2)(\sum_{i=1}^{n} a_i)^2$, instances with more than 200 jobs provoke problems of integer size. We can notice that in that case, the computation time is more important than in cases 1–9, but remains very reasonable.

Conclusions

We investigate in this paper a two-machine open shop scheduling problem, with the assumption that one machine is not always available for jobs processing. More precisely, we assume that the first machine is not available between a time s and a time t and that a job can be interrupted at time s and resumed on the same machine at time t, without any penalty. In the general case with $s > 0$, we show that the problem is ordinary NP-hard, by proposing a pseudo-polynomial dynamic programming algorithm. An integer linear programming formulation of the problem is proposed and we show that any heuristic algorithm for this problem has a worst-case error bound of 1. Computational experiments show that this problem is not difficult to solve in practice, since instances with up to 500 jobs can be solved optimally in less than 5 min by using CPLEX.

More general cases can be investigated now. Problem $F2$, $h_{kj}||C_{\max}$ has been considered,[1] but problems $J2$, $h_{kj}||C_{\max}$ and $O2$, $h_{kj}||C_{\max}$ have never been considered, to our

Table 2 Results of CPLEX solver

	$s \in [0, \frac{1}{3}\sum a_i]$								
	Case 1			*Case 2*			*Case 3*		
n	$\bar{t}$	t_{min}	t_{max}	$\bar{t}$	t_{min}	t_{max}	$\bar{t}$	t_{min}	t_{max}
200	5.800	5.430	7.690	6.133	4.340	11.480	6.044	4.400	9.000
300	14.909	13.510	17.750	14.221	13.350	18.340	15.804	13.510	24.660
400	30.059	24.390	52.780	30.857	24.780	62.180	33.623	24.890	60.530
500	79.961	48.000	140.610	99.201	73.930	173.510	106.164	73.600	245.950
	$s \in [\frac{1}{3}\sum a_i, \frac{2}{3}\sum a_i]$								
	Case 4			*Case 5*			*Case 6*		
n	$\bar{t}$	t_{min}	t_{max}	$\bar{t}$	t_{min}	t_{max}	$\bar{t}$	t_{min}	t_{max}
200	5.231	4.180	5.870	5.244	3.900	5.930	5.282	4.940	5.930
300	12.721	11.690	14.560	12.567	11.590	13.620	12.728	11.750	14.120
400	25.977	23.390	34.000	26.365	22.360	35.810	25.469	23.460	32.240
500	77.202	43.660	123.200	76.904	42.680	126.880	76.995	40.970	122.100
	$s \in [\frac{2}{3}\sum a_i, \sum a_i]$								
	Case 7			*Case 8*			*Case 9*		
n	$\bar{t}$	t_{min}	t_{max}	$\bar{t}$	t_{min}	t_{max}	$\bar{t}$	t_{min}	t_{max}
200	4.687	3.190	5.160	4.647	3.190	5.500	4.619	3.190	5.600
300	11.230	10.320	14.390	11.170	10.490	12.310	11.432	7.360	16.260
400	23.888	13.350	47.230	28.290	19.560	60.200	31.793	15.710	58.500
500	74.150	58.270	192.680	96.915	64.870	170.760	116.942	86.070	269.300

Table 3 Results of CPLEX solver with PARTITION like instances

n	$\bar{t}$	t_{min}	t_{max}
010	0.090	0.000	0.710
020	0.146	0.050	0.280
030	0.223	0.100	0.490
040	0.632	0.220	1.590
050	0.626	0.430	1.320
060	1.109	0.610	4.720
070	1.369	0.820	3.960
080	1.934	1.100	5.440
090	3.300	1.530	17.360
100	3.811	1.760	16.640
125	7.437	2.860	26.640
150	19.711	4.780	56.460
175	20.833	5.440	69.480
200	27.644	10.110	84.860

knowledge. Some other types of unavailability constraints can be considered, like maintenance activities that affect the production rate[16] for instance, or mixed cases with resumable activities together with semi-resumable activities and non-resumable activities.

References

1 Kubiak W, Blazewicz J, Formanowicz P, Breit J and Schmidt G (2002). Two-machine flow shops with limited machine availability. *Eur J Opl Res* **136**: 528–540.

2 Lorigeon T, Bouquard J-L and Billaut J-C (2001). Two machine open shop scheduling problems with availability constraint. *Fifth Workshop on Models and Algorithms for Planning and Scheduling Problems* (*MAPSP'01*). Aussois, France, June 2001.

3 Lee C-Y and Chen Z-L (2000). Scheduling jobs and maintenance activities on parallel machines. *Naval Res Log* **47**: 145–165.

4 Lee C-Y (1997). Minimizing the makespan in the two-machine flow shop scheduling problem with an availability constraint. *Opns Res Lett* **20**: 129–139.

5 Lee C-Y (1999). Two-machine flow shop scheduling with availability constraints. *Eur J Opl Res* **114**: 420–429.

6 Cheng TCE and Wang G (2000). An improved heuristic for two-machine flow shop scheduling with an availability constraint. *Opns Res Lett* **26**: 223–229.

7 Cheng TCE and Wang G (1999). Two-machine flow shop scheduling with consecutive availability constraints. *Inform Process Lett* **71**: 49–54.

8 Blazewicz J, Breit J, Formanowicz P, Kubiak W and Schmidt G (2001). Heuristic algorithms for the two-machine flowshop with limited machine availability. *Omega* **29**: 599–608.

9 Sanlaville E and Schmidt G (1998). Machine scheduling with availability constraints. *Acta Inform* **35**: 795–811.

10 Schmidt G (2000). Scheduling with limited machine availability. *Eur J Opl Res* **121**: 1–15.
11 Blazewicz J and Formanowicz P (2000). Scheduling jobs in open shops with limited machine availability. *Research Report RA-004/2000*. Institute of Computing Science: Poznan.
12 Breit J, Schmidt G and Strusevich VA (2001). Two-machine open shop scheduling with an availability constraint. *Opns Res Lett* **29**: 65–77.
13 Lu L and Posner ME (1993). An NP-hard open shop scheduling problem with polynomial average time complexity. *Math Opl Res* **18**: 12–38.
14 Gonzales T and Sahni S (1976). Open shop scheduling to minimize finish time. *J ACM* **23**: 665–679.
15 Johnson SM (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Res Log Q* **1**: 61–68.
16 Lee C-Y and VJ Leon (2001). Machine scheduling with a rate-modifying activity. *Eur J Opl Res* **128**: 119–128.

Received April 2001;
accepted April 2002 after one revision