MINIMIZING THE MAKESPAN IN TWO-MACHINE JOB SHOP SCHEDULING PROBLEMS WITH NO MACHINE IDLE-TIME

Fatma Hermès
Faculté des Sciences Mathématiques, Physiques et Naturelles de Tunis, Département Informatique, Laboratoire LI3
Campus Universitaire, 1060 Tunis, Tunisia

Jacques Carlier, Aziz Moukrim
Université de Technologie de Compiègne, Centre de Recherches de Royallieu
Laboratoire Heudiasyc, UMR CNRS 6599, BP 20529, 60205 Compiègne cedex, France

Khaled Ghédira
Institut Supérieur de Gestion de Tunis, Laboratoire LI3
Université de Tunis, 41 Rue de la Liberté – Bouchoucha, 2000 Bardo, Tunisia

Keywords: Scheduling, Job Shop, Two Machines, No-idle Constraint, Makespan, Optimal Solution.

Abstract: This paper deals with two-machine job shop scheduling problems working under the no-idle constraint, that is, machines must work continuously without idle intervals. The makespan (C_{max}) has to be minimized. First, we study the problem where each job consists of at most two operations and we show that it can be solved polynomially using Jackson's rule (Jackson, 1956). Second, we study the problem where the number of operations per job can be greater than two and all operations are of unit time and we extend the results of (Hefetz and Adiri, 1982). Finally, we discuss the possibility of getting feasible solutions and then optimal solutions in the general case where the number of operations per job can be greater than two and all operations do not have the same processing time.

1 INTRODUCTION

Frequently, the cost of making machines wait is so high that a no-idle constraint is imposed on machines and no intermediate idle time between operations processed by the same machine is allowed. For example, if the machine is an oven that must cook some pieces at a given high temperature then maintaining the required temperature of the oven while it is empty may be too costly. However, studies of problems on this topic have not attracted a great deal of attention. In the literature, we find some works, most of which are recent, on the permutation flow shop ((Adiri and Pohoryles, 1982), (Baptiste and Lee, 1997), (Kalczynski and Kamburowski, 2007), (Saadani, Guinet and Moalla, 2001), (Saadani, Guinet and Moalla, 2003)). There are also some recent works discussing one machine scheduling problems ((Chrétienne, 2008), (Valente and Alves, 2005), (Valente, 2006)).

The aim of this paper is to study two-machine job shop problems where a set I of n jobs, I = {1, ..., n}, has to be scheduled without intermediate delay on two machines in order to minimize the maximum of the completion times of the jobs i.e. the makespan (C_{max}). Each job i, $i \in I$, is composed of n_i operations $O_{i,j}$, $j = 1 \ldots n_i$, and each operation $O_{i,j}$ has to be processed on a fixed machine for $p_{i,j}$ time units.

The job shop problem plays an important role in the scheduling theory because of its practical applications. Most of job shop problems are NP-hard and there are only few special cases which can be solved polynomially. The two-machine job shop problem with at most two operations per job is denoted $J2|n_i \leq 2|C_{max}$. It was solved polynomially by Jackson (Jackson, 1956) who proposed an algorithm which calculates an optimal schedule in O(n*log(n)) steps using Johnson's rule (Johnson, 1954). The two-machine unit-time job shop problem is denoted

$J2|p_{i,j}=1|C_{max}$. It was proved to be polynomial by (Hefetz and Adiri, 1982) where authors proposed the longest remaining processing time first algorithm which schedules operations in a decreasing order of the remaining processing time of jobs. Lenstra, Rinnooy Kan and Brucker showed in (Lenstra, Rinnooy Kan and Brucker, 1977) that problem $J2||C_{max}$ is strongly NP-hard. Later, Brucker showed in (Brucker, 1994) that the two-machine job shop problem with a fixed number k of jobs, denoted $J2|n=k|C_{max}$, can be solved polynomially by reducing it to a shortest path problem and then he deduced that it is possible to calculate an optimal schedule for $J2||C_{max}$ for any fixed number of jobs in polynomial time.

The paper is organized as follows: In section 2, we define the studied problem and we recall some backgrounds and basic results relying on the two-machine job shop problem. In section 3, we study the two-machine job shop problem where the number of operations per job is at most equal to two and machines must work under the no-idle constraint. This problem is denoted $J2|n_i\leq 2,\text{no-idle}|C_{max}$. We show that it can be solved in polynomial time using Jackson's rule (Jackson, 1956). In section 4, we study the two-machine unit-time job shop problem with no machine idle time, denoted $J2|p_{i,j}=1,\text{no-idle}|C_{max}$, and we extend the results of (Hefetz and Adiri,1982). Finally, in section 5, we deduce some special cases which are polynomially solvable.

2 GENERAL POINTS

In this section, we first define the problem subject of this study and we present some definitions. Next, we present Johnson's and Jackson's algorithms where Johnson's algorithm (Johnson, 1954) solves the two-machine flow shop problem with C_{max} criterion denoted $F2||C_{max}$ and Jackson's algorithm (Jackson, 1956) solves problem $J2|n_i\leq 2|C_{max}$ using Johnson's rule. Finally we introduce the longest remaining processing time first algorithm (Hefetz and Adiri, 1982) which solves problem $J2|p_{ij}=1|C_{max}$.

2.1 Problem Formulation and Basic Definitions

The two-machine job shop problem is a problem where a set I of n jobs, I = {1, ..., n} have to be processed in a shop with two machines M_1 and M_2. Each job i, i = 1, …, n, consists of a sequence of n_i operations $O_{i,1}$, $O_{i,2}$, ..., $O_{i,ni}$ which must be processed in this order. The precedence constraints are so that $O_{i,j}$ precedes $O_{i,j+1}$, j = 1, …, $n_i - 1$. Each operation $O_{i,j}$ must be processed for $p_{i,j}$ time units on machine $\mu_{i,j} \in \{M_1, M_2\}$.

The following assumptions are made:

- A machine can process only one operation at a time.
- An operation cannot be interrupted.
- The time zero is the earliest time an operation can be started.
- All setup times are included into the job processing times.
- If operation $O_{i,j}$ must be processed on machine M_1, then operation $O_{i,j+1}$ must be processed on machine M_2 ($\mu_{i,j} \neq \mu_{i,j+1}$ for i = 1,…, $n_i - 1$). Thus, job i may be characterized by the number of operations and the machine on which the first operation must be processed.
- Only no-idle schedules are considered.

Let $t_{i,j}$ be the starting time of operation $O_{i,j}$ and let $C_{i,j}$ be its completion time. Let C_i be the completion time of job i so that

$$C_i = \max_{j=1\ldots n_i} C_{i,j} \tag{1}$$

Let us present the following definitions:

- An *initial operation* is one without predecessors: the operation $O_{i,1}$ is the initial operation for job i.
- A *terminal operation* is one without successors: the operation O_{i,n_i} is the terminal operation for job i.
- A *ready operation* is an operation that has not yet been scheduled while all its predecessors have been.
- A *no-idle schedule* satisfies the no-idle constraint on each machine. In other words, if operation $O_{i,j}$ is executed immediately before operation $O_{i',j'}$ on the same machine then we have:

$$C_{i,j} = t_{i,j} + p_{i,j} = t_{i',j'} \tag{2}$$

Given a feasible schedule π we have:

$$C_{max}(\pi) = \max_{\substack{i=1\ldots n \\ j=1\ldots n_i}} C_{i,j} = \max_{i=1\ldots n} C_i \tag{3}$$

The objective is to find a no-idle schedule so as to minimize the C_{max}. The problem so formulated is denoted $J2|\text{no-idle}|C_{max}$. When the number of operations per job is at most equal to two, the problem is denoted $J2|n_i\leq 2,\text{no-idle}|C_{max}$ and when all operations have the same processing time which is considered as the unit-time, the problem is denoted $J2|p_{i,j}=1,\text{no-idle}|C_{max}$.

2.2 Jackson's and Johnson's Algorithms

Jackson's algorithm constructs an optimal schedule for $J2|n_i \le 2|C_{max}$ problem reducing it to $F2||C_{max}$ problem. Below, we first describe Johnson's algorithm. Then, we present Jackson's algorithm. Finally, we conclude with some results concerning the presence of idle times on machines.

2.2.1 Johnson's Algorithm

The two-machine flow shop problem is a problem where a set I' of n' jobs, $I' = \{1,2,\dots,n'\}$ must visit machines in the same order. Each job i consists of two operations $O_{i,1}$ and $O_{i,2}$ which must be processed respectively, first on machine M_1 then on machine M_2. Johnson's algorithm (Johnson, 1954) constructs an optimal schedule in polynomial time $(O(n*\log(n)))$ for $F2||C_{max}$ problem. It applies the following steps:

i. Divide the set of jobs I', $I' = \{1,2,\dots,n'\}$, into two subsets:
   a. Let I'_1 denote the subset of jobs i, i = 1, …, n', which satisfy the condition $p_{i,1} \le p_{i,2}$
   b. Let I'_2 denote the subset of jobs i, i = 1, …, n', which satisfy the condition $p_{i,1} > p_{i,2}$.

ii. Schedule on each machine first the jobs of I'_1 in an increasing order of $p_{i,1}$ and then the jobs of I'_2 in a decreasing order of $p_{i,2}$.

2.2.2 Jackson's Algorithm

Jackson's algorithm (Jackson, 1956) calculates an optimal solution for problem $J2|n_i \le 2|C_{max}$, in polynomial time $(O(n*\log(n)))$ by first reducing it to $F2||C_{max}$ problem and then using Johnson's rule. It applies the following steps:

i. Divide the set of jobs I = {1, 2, …, n} into four subsets:
   a. Let I_1 denote the subset of jobs consisted of only one operation which must be processed on machine M_1.
   b. Let I_2 denote the subset of jobs consisted of only one operation which must be processed on machine M_2.
   c. Let $I_{1,2}$ denote the subset of jobs which are processed first on machine M_1 then on machine M_2.
   d. Let $I_{2,1}$ denote the subset of jobs which are processed first on machine M_2 then on machine M_1.

ii. Calculate an optimal sequence $R_{1,2}$ for the flow shop problem relative to the job set $I_{1,2}$.

iii. Calculate an optimal sequence $R_{2,1}$ for the flow shop problem relative to the job set $I_{2,1}$.

iv. On machine M_1 schedule first $I_{1,2}$ according to $R_{1,2}$, then all jobs in I_1 and finally $I_{2,1}$ according to $R_{2,1}$.

v. On machine M_2 schedule first $I_{2,1}$ according to $R_{2,1}$, then all jobs in I_2 and finally $I_{1,2}$ according to $R_{1,2}$.

2.2.3 Further Results

Adiri and Pohoryles (Adiri and Pohoryles, 1982) observe that problems $F2|prmu,no\text{-}idle|C_{max}$ and $F2|prmu|C_{max}$ are equivalent in the sense that every $F2|prmu|C_{max}$ schedule can be transformed into an $F2|prmu,no\text{-}idle|C_{max}$ schedule with maintaining the same C_{max}. Thus, both problems can be solved by Johnson's algorithm (Johnson, 1954). Johnson's schedule is an active schedule in which machine M_1 is naturally no-idle since operations are processed consecutively on it without idle interval. Moreover, the jobs preceding each idle interval on the second machine M_2 can be delayed without increasing the C_{max}. It is enough to fix the starting time of the last operation scheduled on M_2 and to schedule the other operations so that all operations are scheduled consecutively without any intermediate delay.

Brucker (Brucker, 1995) observes that in Jackson's schedule at least one machine processes jobs without idle intervals. More specifically, having the following assumption:

$$\sum_{i \in I_{2,1}} p_{i,2} \le \sum_{i \in I_{1,2}} p_{i,1} + \sum_{i \in I_1} p_{i,1} \qquad (4)$$

then there is no idle time on machine M_1. Otherwise, there is no idle time on machine M_2.

2.3 The Longest Remaining Processing Time First Algorithm

The longest remaining processing time first algorithm has been proposed by (Hefetz and Adiri, 1982) to solve $J2|p_{i,j}=1|C_{max}$ problem. It constructs an optimal schedule for this problem with applying the following steps:

i. Give a label $\alpha_{i,j}$ to each operation $O_{i,j}$ so that:

$$\alpha_{i,j} = n_i - j + 1 \qquad i = 1,\dots,n \text{ and } j = 1,\dots,n_i \qquad (5)$$

ii. Schedule the highest label operation for the earliest possible time on the required machine, with ties broken arbitrarily.

iii. Remove from the problem the scheduled operation. Stop if all operations are scheduled, otherwise return to (ii).

The authors noted that the operation with the highest label in step (ii) must be a ready operation, since, if it is not then there is an unscheduled predecessor with a higher label, which is a contradiction.

Let T_j, j = 1, 2, be the total processing time required on machine M_j and p_i the processing time of job i. Thus, $p_i = n_i$ in view of the fact that all operations have unit duration.

Theorem 1 (Hefetz and Adiri, 1982). The longest remaining processing time first algorithm constructs an optimal schedule for $J2|p_{i,j}=1|C_{max}$ problem. If all initial operations require the same machine and we have:

$$T_1 = T_2 \geq \max_i n_i \tag{6}$$

Then, the optimal schedule length is:

$$C^*_{max} = T_1 + 1 = T_2 + 1 \tag{7}$$

Otherwise, the optimal schedule length is:

$$C^*_{max} = \max(T_1, T_2, \max_i n_i) \tag{8}$$

3 RESOLUTION OF THE PROBLEM J2|NI≤2,NO-IDLE|CMAX

The problem we consider in this section is to find an optimal schedule for $J2|n_i\leq 2$,no-idle$|C_{max}$ problem. We propose first to study the feasibility of Jackson's schedule then, we prove its optimality. We also deduce some interesting results which concern the necessity of applying Johnson's rule in Jackson's algorithm.

Proposition 1. The C_{max}-value of Jackson's schedule is a lower bound for the optimal C_{max} of $J2|n_i\leq 2$,no-idle$|C_{max}$ problem.

Proof. Jackson's schedule is an active schedule in which all the operations are scheduled as soon as possible. On the other hand, to satisfy the no-idle constraint, some operations must be delayed which must increase the C_{max}-value. So, the optimal C_{max} of $J2|n_i\leq 2$,no-idle$|C_{max}$ problem must be greater than or equal to that of $J2|n_i\leq 2|C_{max}$ problem.

So, evidently Jackson's schedule is optimal for $J2|n_i\leq 2$,no-idle$|C_{max}$ problem if it is no-idle or if it can be transformed into a no-idle schedule without increasing the C_{max}-value.

Lemma 1. If (4) then there is no-idle time on machine M_1 and it is unnecessary for Jackson's algorithm to apply Johnson's rule for the two-machine flow shop sub-problem relative to $I_{2,1}$.

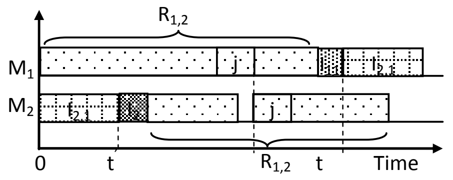

Figure 1: A schedule where there is no-idle time on machine M_1.

Proof. If (4) holds, then at time t where,

$$t = \sum_{i\in I_{1,2}} p_{i,1} + \sum_{i\in I_1} p_{i,1} \tag{9}$$

all the operations of $I_{2,1}$ which must be processed on machine M_1 are ready. Therefore, these operations can be processed without idle time immediately after the end of those of I_1 as presented in figure 1 above.

As a result, the order of processing the operations of $I_{2,1}$ on machine M_1 and on machine M_2 does not affect the C_{max}-value since all the operations of $I_{2,1}$ which must be processed on machine M_2 are naturally ready at time 0. They are completed at time t', where

$$t' = \sum_{i\in I_{2,1}} p_{i,2} \leq t \tag{10}$$

Lemma 2. If

$$\sum_{i\in I_{1,2}} p_{i,1} \leq \sum_{i\in I_{2,1}} p_{i,2} + \sum_{i\in I_2} p_{i,2} \tag{11}$$

then there is no-idle time on machine M_2 and it is unnecessary for Jackson's algorithm to apply Johnson's rule for the two-machine flow shop sub-problem relative to $I_{1,2}$.

Proof. The proof is similar to that of lemma 1.

Let us note that if

$$\sum_{i\in I_{2,1}} p_{i,2} \leq \sum_{i\in I_{1,2}} p_{i,1} \tag{12}$$

then (4) holds. Also, if

$$\sum_{i\in I_{1,2}} p_{i,1} \leq \sum_{i\in I_{2,1}} p_{i,2} \tag{13}$$

then (11) holds.

By summation of (12) and (13), we have:

$$\sum_{i\in I_{1,2}} p_{i,1} = \sum_{i\in I_{2,1}} p_{i,2} \tag{14}$$

Lemma 3. If any of the following assumptions hold, namely, (4) and (13), (11) and (12), or (14) then Jackson's schedule is no-idle. It is consequently optimal for problem $J2|n_i\leq 2$,no-idle$|C_{max}$ and it is unnecessary for Jackson's algorithm to apply Johnson's rule neither for the two-machine flow shop sub-problem relative to $I_{1,2}$ nor for the two-machine flow shop sub-problem relative to $I_{2,1}$.

Proof. If (4) holds then there is no idle time on machine M_1 and it is pointless to apply Johnson's rule in Jackson's algorithm to the sub-problem relative to $I_{2,1}$ (lemma 1).

If (13) holds then there is no-idle time on machine M_2 and it is pointless to apply Johnson's rule in Jackson's algorithm to the sub-problem relative to $I_{2,1}$ (lemma 2).

By summation of (4) and (13), Jackson's schedule is no-idle. It is consequently optimal for $J2|n_i\leq 2$,no-idle$|C_{max}$ problem and it is unnecessary for Jackson's algorithm to apply Johnson's rule neither for the two-machine flow shop sub-problem relative to $I_{1,2}$ nor for the two-machine flow shop sub-problem relative to $I_{2,1}$.

In the same way, we deduce the same results if assumptions (11) and (12) hold or assumption (14) holds.

Proposition 2. The problem $J2|n_i\leq 2$,no-idle$|C_{max}$ is polynomial. It can be solved with Jackson's algorithm.

Proof. At least one machine is no-idle (Brucker, 1995). If there is an idle interval on machine M_1 then it is between the operations of $I_{2,1}$ and in this case the assumption (4) does not hold. If there is an idle interval on machine M_2 then it is between the operations of $I_{1,2}$ and in this case the assumption (11) does not hold either.

Supposing that machine M_2 contains an idle time, then this idle time is between the operations of $I_{1,2}$ relative to the two-machine flow shop sub-problem where each job visits first, machine M_1 then machine M_2. This idle time can be reduced by fixing the starting time of the last operation scheduled on machine M_2 and scheduling the other operations of $I_{1,2}$ consecutively without any intermediate delay (Adiri and Pohoryles, 1982).

This action creates an other idle time between the first operation scheduled of $I_{1,2}$ on machine M_2 and the last operation scheduled of I_2 on the same machine.

The last idle time can also be reduced by delaying all the operations of I_2 and then the idle time becomes between the last operation scheduled of $I_{2,1}$ and the first operation scheduled of I_2. Since, there is no idle time on machine M_1, then (4) holds.

Let t be the ending date of the last operation of $I_{1,2}$ scheduled on machine M_1 and let t' be the starting time of the first operation of $I_{1,2}$ scheduled on machine M_2.

Naturally, we have:

$$t' \leq t \tag{15}$$

Because if (15) does not hold, then (4) does not hold. Consequently, it is possible to shift the operations of $I_{2,1}$ on machine M_2 to the right so that there is no-idle time on machine M_2.

Thus, the schedule constructed by Jackson's algorithm can be easily transformed into a no-idle schedule without increasing the C_{max}-value. It is enough to fix the last operation on the machine which contains idle intervals and shift to the right all the other operations in order to have no idle intervals.

As a result, Jackson's algorithm also constructs an optimal schedule for he problem $J2|n_i\leq 2$,no-idle$|C_{max}$ problem.

Proposition 3. The set of optimal solutions of $J2|n_i\leq 2$,no-idle$|C_{max}$ problem is included in the set of optimal solutions of $J2|n_i\leq 2|C_{max}$ problem.

Proof. Jackson's schedule can easily be transformed to a no-idle schedule without increasing the C_{max}-value. So, both problems have the same C_{max}^*. Besides, an optimal solution for $J2|n_i\leq 2$,no-idle$|C_{max}$ problem is also optimal for $J2|n_i\leq 2|C_{max}$ problem.

Thus, the set of optimal solutions of $J2|n_i\leq 2$,no-idle$|C_{max}$ problem is included in the set of optimal solutions of $J2|n_i\leq 2|C_{max}$ problem.

4 RESOLUTION OF THE PROBLEM $J2|p_{ij}=1$,no-idle$|C_{max}$

In this section, we study $J2|p_{ij}=1$,no-idle$|C_{max}$ problem.

Proposition 4. The optimal C_{max} of $J2|p_{ij}=1|C_{max}$ problem is a lower bound for the optimal C_{max} of $J2|p_{ij}=1$,no-idle$|C_{max}$ problem; if an optimal schedule exists.

Proof. The proof is similar to that of proposition 1.

Below, we denote HA the longest remaining processing time first algorithm. Let S be the schedule constructed by HA algorithm.

Lemma 4. If there is no-idle time in S then the makespan of S is minimal and the number of idle times is minimal.

Proof. HA calculates a schedule with minimal makespan for $J2|p_{ij} = 1|C_{max}$ problem (theorem 1). Moreover, if there is no-idle time in S then the number of idle times is also minimal.

Let us introduce a second criterion to minimize which is the number of idle times denoted $\bar{A}$. The objective is then to minimize first the C_{max} then $\bar{A}$. In this case, the related problem is denoted $J2|p_{ij}=1|Lex(C_{max},\bar{A})$.

Evidently, if an optimal schedule for $J2|p_{ij}=1|Lex(C_{max},\bar{A})$ problem is no-idle then it is also optimal for $J2|p_{ij}=1,\text{no-idle}|C_{max}$ problem. Also, if there is no-idle time in S, then S is optimal for both problems $J2|p_{ij}=1,\text{no-idle}|C_{max}$ and $J2|p_{ij}=1|Lex(C_{max},\bar{A})$. Furthermore, if it is possible to built a no-idle schedule S' from S without increasing the C_{max} then S' is optimal for both problems $J2|p_{ij}=1,\text{no-idle}|C_{max}$ and $J2|p_{ij}=1|Lex(C_{max},\bar{A})$.

Let us assume that job h is the job so that

$$n_h = \max_i n_i \tag{16}$$

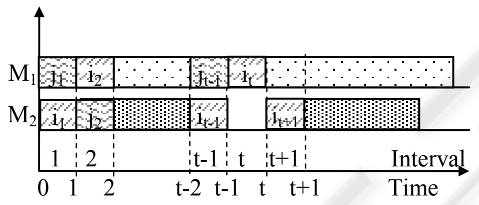

Figure 2: The schedule resulting from the assumption of an idle interval on machine M_2 in interval t and t is even.

Lemma 5. If in S there is an idle time, on some machine, then

i. the job h is unique, and it is processed continuously from time 0 to n_h, alternating on the two machines;
ii. the only job processed on the machine containing this idle time is the job h;
iii. from time t – 1, all the operations processed on the other machine, except of those of the job h, must have a label 1.

Proof. The time axis is supposed to be split into intervals of unit times. We suppose that the first idle interval is the interval t. Let us suppose also that t is even and that this idle time is on machine M_2. Let us denote i_1 (resp: j_1) the job processed in interval 1 on machine M_2 (resp: M_1) and so on. So, i_{t-1} is the job processed on machine M_2 in interval t – 1, and i_t is the job processed on machine M_1 in interval t (figure 2). The label of job j_{t-1} is 1. Otherwise there doesn't exist any idle time in interval t. Consequently, all jobs processed after t – 1, at the exception of job i_t which is the only job which can have a label strictly larger than 1. Let us set $i = i_t$. We prove that $i_{t-1} = i$. It is the only job which can have a label strictly larger than 2. For the same reason $i_{t-2} = i$ (the only job which can have a label strictly larger than 3) and so on until $i_1= i$. Consequently, i = h. The same reasoning can be applied when t is odd.

Thus, the job i must be the job h and then the job h is continuously scheduled from beginning to end since at each time it have the greatest label. It is consequently the job h verifying the assumption (16) and it is unique.

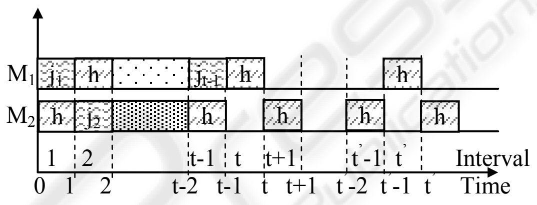

Figure 3: The first operation of job h is processed on machine M_2 and machine M_1 work for the first interval.

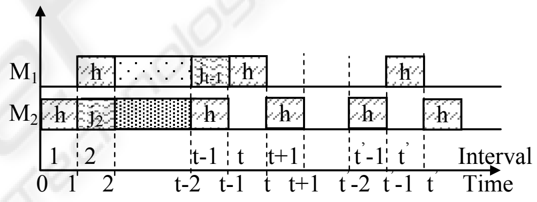

Figure 4: The first operation of job h is processed on machine M_2 and machine M_1 is free for the first interval.

Algorithm IT

i. Built S with applying HA algorithm
ii. Built S' by scheduling the first operation of job h at time 0 and letting the other machine idle at time 0 and scheduling the remaining jobs by applying HA algorithm from time 1.

Theorem 2. One of the two schedules built by algorithm IT is optimal for both objective functions.

Proof. First, let us note that if there is an idle time in S then this idle time usually precedes an operation of job h and it is also preceded with an operation of the same job h. Indeed, if there is an idle time then the job h is unique and therefore it is continuously scheduled from beginning to the end (lemma 5). Thus, in each interval the job h is processed on one machine and on the other machine either another job is processed or there is an idle time. On the other hand, if the first operation of job h is processed on machine M_2 then if there is an idle time on machine M_2 which occurs in interval t then t is even and if there an idle time on machine M_1 which occurs in interval t' then t' is odd. Otherwise, t is odd and t' is even.

Below, we show in four cases that S or S' is optimal for both objectives.

In the first case, we assume that the first operation of job h is processed on machine M_2 and the machine M_1 works for the first interval of time (figure 3). In this case, the number of idle times on machine M_2 is minimal because the operations scheduled on machine M_2, except of those of the job h, are used optimally to fill idle intervals created by the job h. It is the same for machine M_1 from time 1. However, if it is possible to make machine begin working later with one unit of time without increasing the ending date on this machine and the C_{max} then we can suppress an idle time. This is done eventually in S'.

In the second case, we assume that the first operation of job h is processed on machine M_1 and the machine M_2 works for the first interval of time. This case is similar to the first one.

In the third case, we assume that the first operation of job h is processed on machine M_2 and the machine M_1 is free for the first interval of time (figure 4). In this case, the number of idle times is minimal in both machines because in each machine the first operation scheduled is an operation of job h.

In the fourth case, we assume that the first operation of job h is processed on machine M_1 and the machine M_2 is free for the first interval of time. This case is similar to the third one.

Thus, S or S' is optimal for both objectives.

We can resume that S is no-idle if and only if one of the following cases holds:

Case 1:

$$\max_i n_i \leq \min(T_1, T_2) \quad (18)$$

Case 2:

$$\max_i n_i = \min(T_1, T_2) + 1 \quad (19)$$

and the last operation of the job h is scheduled on the machine which determines the schedule length.

Case 3:

$$\max_i n_i = \min(T_1, T_2) + 2 \quad (20)$$

and all jobs begin in the same machine and the last operation of the job h is scheduled on the machine which determines the schedule length.

Finally, we note that S' is no-idle if S is no-idle or S contains only one idle time.

5 RESOLUTION OF THE PROBLEM J2|n=k, no-idle|C_{max}

In this section, we discuss J2|n=k,no-idle|C_{max} problem.

Proposition 5. The optimal C_{max} of J2|n=k|C_{max} problem is a lower bound for the optimal C_{max} of J2|n=k,no-idle|C_{max} problem; if an optimal schedule exists.

Proof. The proof is similar to that of proposition 1.

So, the optimal schedule for J2|n=k|C_{max} problem is optimal for J2|n=k,no-idle|C_{max} problem if it is no-idle or if it can be transformed into no-idle schedule without increasing the C_{max}-value.

Obviously, if n = 1 and $n_1 > 2$ then it is not possible to construct a no-idle schedule. There are exactly $n_1 - 2$ idle intervals.

Let us consider the problem where the number of jobs is equal to two. This problem is denoted J2|n=2,no-idle|C_{max}.

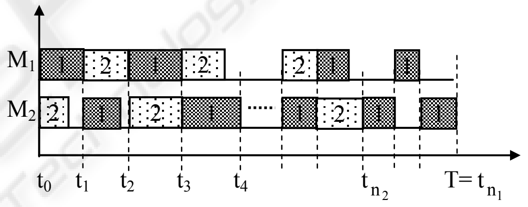

Figure 5: Schedule format with two jobs where $n_1 > n_2$.

Let us give the following cases:

Case 1: $n_1 = n_2$, $\mu_{1,1} \neq \mu_{2,1}$ and $p_{1,j} = p_{2,j}$ for $j = 2\ldots n_1 - 1$

Case 2: $n_1 = n_2$, $\mu_{1,1} = \mu_{2,1}$ and $p_{1,j+1} = p_{2,j}$ for $j = 1\ldots n_1 - 1$

Case 3: $n_1 = n_2$, $\mu_{1,1} = \mu_{2,1}$ and $p_{1,j} = p_{2,j+1}$ for $j = 1\ldots n_1 - 1$

Case 4: $n_1 = n_2 + 1$, $\mu_{1,1} \neq \mu_{2,1}$ and $p_{1,j} = p_{2,j}$ for $j = 2\ldots n_2$

Case 5: $n_1 = n_2 + 1$ and $\mu_{1,1} = \mu_{2,1}$ and $p_{1,j+1} = p_{2,j}$ for $j = 1\ldots n_2 - 1$

Case 6: $n_2 = n_1 + 1$, $\mu_{1,1} \neq \mu_{2,1}$ and $p_{1,j} = p_{2,j}$ for $j = 2\ldots n_1$

Case 7: $n_2 = n_1 + 1$, $\mu_{1,1} = \mu_{2,1}$ and $p_{1,j} = p_{2,j+1}$ for $j = 1\ldots n_1 - 1$

Case 8: $n_1 = n_2 + 2$, $\mu_{1,1} = \mu_{2,1}$, and $p_{1,j+1} = p_{2,j}$ for $j = 1\ldots n_2$

Case 9: $n_2 = n_1 + 2$, $\mu_{1,1} = \mu_{2,1}$, and $p_{1,j} = p_{2,j+1}$ for $j = 1\ldots n_1$

Proposition 6. If any of the previous cases holds then the no-idle schedule is unique; it is consequently the optimal solution for J2|n=2,no-idle|C_{max} problem. Otherwise, it is impossible to get a no-idle schedule.

Proof. A feasible schedule for J2|n=2|C_{max} problem takes the format presented in figure 5 above. Clearly, having this format, it is not possible to transform any schedule to a no-idle schedule. There are at most max(n_1, n_2) – 2 idle intervals.

However, we deduce that if any of the previous cases holds then the no-idle schedule is unique. It is consequently the optimal solution for J2|n=2,no-idle|C_{max} problem. Otherwise, it is impossible to get a no-idle schedule.

6 CONCLUSIONS

In this paper, we have studied the impact of adding the no-idle constraint to the problem of minimizing the makespan in a two-machine job shop. We have studied separately the case where the number of operations per job isn't greater than two and the case where all operations are of unit time. In the first case we have showed that there exists usually an optimal schedule which we can calculate using Jackson's rule and then fixing the last operation scheduled on the machine which contains an idle time and then scheduling the other operations consecutively without idle times. However, in the second case, we showed that it is not usually possible to build a feasible no-idle schedule. Then, we have proposed the IT algorithm which minimizes first the C_{max} then the number of idle times (Ā). We have shown that it is impossible to build a schedule which contains a number of idle times smaller than that of the schedule obtained by applying IT algorithm. Consequently, if this schedule is no-idle then it is also optimal for the corresponding problem with adding the no-idle constraint. Moreover, in the general case, where the number of operations per job can be greater than two and all operations do not have the same processing time, we have shown that where the number of jobs is equal to two there are only few cases numbered from 1 to 9 which are efficiently solvable and where the set of feasible no-idle schedules contains a unique schedule. In conclusion, we deduce that it is not usually possible to construct a feasible no-idle schedule for the two-machine job shop problem and that in the majority of cases, this set is empty.

REFERENCES

Adiri, I., Pohoryles, D., 1982. Flow-shop/no-idle or no-wait scheduling to minimise the sum of completion times. *Naval Research Logistics Quarterly*, Vol. 29, pp. 495-504.

Baptiste, P., Lee, K.H., 1997. A branch and bound algorithm for the F|no-idle|C_{max}. *Proceedings of the International Conference on Industrial Engineering and Production Management (IEPM'1997)*, Lyon, vol. 1, pp. 429-438.

Brucker, P., 1994. A polynomial algorithm for the two machine job-shop scheduling problem with fixed number of jobs. *Operations Research Spektrum*, Vol. 16, pp. 5-7.

Brucker, P., 1995. Scheduling Algorithms, *Springer*, ISBN: 3-540-60087-6.

Chrétienne, P., 2008. On single-machine scheduling without intermediate delays. *Discrete Applied Mathematics, 156,* p. 2543-2550.

Garey, M.R.D., Johnson, D.S., Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, Vol. 1, pp. 117-129.

Hefetz, N. , Adiri, I., 1982. An Efficient Optimal Algorithm for the Two-Machines Unit-Time Jobshop Schedule-Length Problem. *Mathematics of Operations Research*, Vol. 7, No. 3., pp. 354-360.

Jackson, J.R., 1956. An extension of Johnson's results on job lot scheduling. *Naval Research Logistic Quarterly*, Vol. 3, pp. 201-203.

Johnson, S.M., 1954. Optimal two- and three-stage production schedules with setup times included, *Naval Research Logistics Quarterly*, Vol. 1, pp. 61-68.

Kalczynski, P. J., Kamburowski, J., 2007. On no-wait and no-idle flow shops with makespan criterion. *European Journal of Operational Research*, Vol. 178, pp. 677–685.

Lenstra, J. K., Rinnooy Kan, A.H.G., Brucker, P., 1977. Complexity machine scheduling problems. *Annals of Discrete Mathematics*, Vol. 1, pp. 343-362.

Saadani, H., Guinet, A., Moalla, M., 2001. A travelling salesman approach to solve the *F*|no-idle|C_{max} problem. *Proceedings of the International Conference on Industrial Engineering and Production Management (IEPM' 2001)*, Quebec, vol. 2, 880-888.

Saadani, H., Guinet, A., Moalla, M., 2003. Three stage no-idle flow-shops. *Computers and Industrial Engineering*, 44, 425-434.

Valente, J. M. S., Alves, R. A. F. S., 2005. Improved Heuristics for the Early/Tardy scheduling problem with no idle time. *Computers & Operations Research*, 32: p. 557-569.

Valente, J. M. S., 2006. Heuristics for the single machine scheduling problem with early and quadratic tardy penalities. Working paper 234, Faculdade de Economia do Porto, Portugal, December.