

Job Shop Scheduling Problem with Multi-purpose Machines and availability constraints

Nozha Zribi Abdelkader EL Kamel Pierre Borne
LAGIS - Ecole Centrale de Lille

January 21, 2005

Abstract. In this paper, we deal with the Job shop scheduling problem with Multi-Purpose Machine and Availability Constraints. First, we propose a polynomial algorithm solving exactly the problem for two jobs only. Then a lower bound as well as a heuristic approach based on this algorithm are proposed for the general problem.

Keywords: job-shop, flexibility, geometric approach, complexity

1. Introduction

In the scheduling literature it is generally assumed that machines are available during the whole planning horizon. However, in many real situations, machines may be non-available for processing jobs after a breakdown or during a preventive maintenance activity. In fact, in manufacturing systems, machines are periodically submitted to maintenance operations. This paper deals with the MPM job-shop with limited machine availability. We consider the deterministic model where the unavailability periods corresponding to maintenance tasks are known in advance. We also assume that preemption of operations is not allowed. More precisely, an operation O_{ij} of job J_i on machine M_k starts only if its execution can be finished before M_k becomes unavailable. The problem we consider is a generalization of the classical job-shop problem and the multi-purpose machine problem studied in (Jurisch, 1992), where machines are available all time.

As compared to the literature dedicated to classical scheduling problems, studies dealing with limited machine scheduling problems are rather rare. Availability constraints have been firstly introduced for a single machine (Adiri et al., 1986) and parallel machines (Schmidt, 1988), (Schmidt, 1988). Lee extensively investigated flow-shop scheduling problems with two machines (Lee, 1996), (Lee, 1997), (Lee, 1999). In particular, the author defined the resumable, non-resumable and semi-resumable models. An operation is called resumable if it can be interrupted by an unavailability period and completed without penalty as soon as the machine becomes available again. If the part of the operation that has been processed before the unavailability period must be partially (respectively fully) re-executed, then the operation is called

semi-resumable (respectively. non-resumable). Recently, flow-shop scheduling problems with two machines and resumable jobs were treated in (Blazewicz et al., 2001), and (Kubiak et al., 2002). Job-shop problem under unavailability constraints has also been considered recently. In (Aggoune, 2002) the author proposed an extension of the geometric approach to solve the two-job shop scheduling problem with availability constraints as well as a branch and bound algorithm with lower bound based on two-job for the job-shop problem with heads and tails and unavailability periods. However to our knowledge MPM Job shop with availability constraints has not been considered yet. The problem is strongly NP-hard since problem without unavailability periods is already strongly NP-hard (Jurisch, 1992). Therefore we propose in this paper an approximation method to solve this problem.

The remainder of this paper is organized as follows. After a description of the considered problem in the following section, we focus, in section 3, on the MPM job shop scheduling problem with two jobs. A polynomial algorithm, which is an extension of the temporized geometric approach introduced by (Aggoune, 2002), is presented. In Section 4 we propose a lower bound for the general problem as well as an heuristic solution.

2. problem description

The MPM job-shop with availability constraints may be formulated as follows. There are n jobs J_1, \dots, J_n to be processed on a set of m machines $\mathfrak{R} = (M_1, \dots, M_m)$. Each machine M_k can process at most one job at a time. Each job J_i consists of a sequence of n_i operations, that must be accomplished according to its manufacturing process. Each operation O_{ij} ($i = 1, \dots, n; j = 1, \dots, n_i$) can be performed by any machine r in a given set $\mu_{ik} \subset \mathfrak{R}$ for p_{ij} time units. The operation is non-preemptive, i.e. it must be accomplished without interruption. Moreover, we assume that machine M_k is unavailable during giving periods corresponding to preventive maintenance. The starting times and durations of these tasks are known and fixed in advance. The objective is to construct a schedule with a minimum makespan.

According to the terminology concerning the machine availability introduced in (Schmidt, 2000), the studied problem can be denoted by $J(MPM), NCwin | Cmax$, where NCwin means that non-availability periods are arbitrarily distributed on machines.

The scheduling problem in $J(MPM)NCwin | C_{max}$ can be decomposed in two subproblems:

- a routing subproblem that consists in assigning operations to machines;
- an operation scheduling subproblem associated with each machine aiming to minimize the makespan. This is a Job-Shop scheduling Problem with Availability Constraints $J(MPM), NCwin | C_{max}$.

3. A Geometric representation for

$$J(MPM)NCwin | n = 2 | C_{max}$$

We focus in this section on the MPM job shop scheduling problem with availability constraints in the particular case of two jobs to be scheduled. We first give a description of the Temporized Geometric Approach (TGA) which is a polynomial algorithm for the two-job shop scheduling problem with limited machine availability. It has been introduced by (Aggoune, 2002) and is an extension of the classical geometric approach developed for the two-job scheduling problem. Then we present a generalization of this approach, which allows dealing with limited machine availability and flexibility of machines.

3.1. TEMPORIZED GEOMETRIC APPROACH FOR CLASSICAL JOB SHOP WITH AVAILABILITY CONSTRAINTS

The geometric approach has been firstly introduced in (Akers and Friedman , 1955). It consists in reducing the two-job shop scheduling problem in the search of a shortest path and thus gives a polynomial algorithm to solve the problem.

In this method the problem can be represented by a two dimension plane with obstacles, defined as follows:

- Each axis representing one job $J_i = O_{i1}, O_{i2}, \dots, O_{in_i}$ is decomposed into n_i sub-intervals. The length L_{ij} of subinterval I_{ij} is proportional to the processing time p_{ij} of operation O_{ij} ,
- Intervals O_{1j} and O_{2k} form an obstacle if O_{1j} and O_{2k} share the same machine (see Fig. 1),
- the rectangle defined by the origin point O and the final point F, which corresponds to the completion of the two jobs, is the final obstacle.

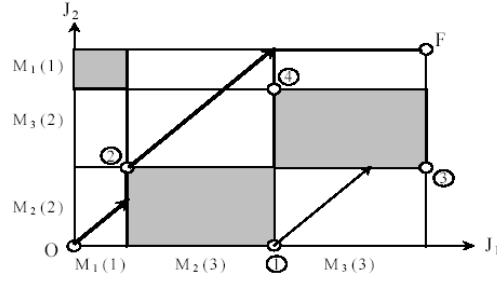


Figure 1. A shortest path in the plane

A feasible solution of the scheduling problem is then a path going from the origin O to the point F . Such a path consists of horizontal, vertical and diagonal legs. A horizontal (resp. vertical) leg represents the exclusive progression of job J_1 (resp. J_2), whereas diagonal legs correspond to simultaneous executions of the two jobs. Moreover, any path must avoid the interior of the obstacles. This is due to the fact that two operations can not be executed simultaneously on the same machine and are not preemptable.

The TGA is an extension of the geometric approach which exactly solve the problem $J, N_{Cwin} | n = 2 | Cmax$. It allows to integrate the evolution of time and so the availability of the machines, based on the definition and the introduction of new vertices, as well as a new and dynamic way to progress from one vertex to its successors.

Vertices Characterization and Definitions:

In the classical geometric approach, vertices of the network are the north-west (NW) and south-east (SE) corners of the obstacles hit when going diagonally in the plane. These corners are located at the extremities of the intervals corresponding to operations in conflict. Each vertex can then be defined thanks to its coordinates in the plane: the x-coordinate (resp. y-coordinate) of the vertex corresponds to operation of job J_1 (resp. J_2) to be executed. In TGA, some vertices can be located between two lines bounding an operation that is to say inside intervals. For each coordinate of the vertices, an additional information related to the duration already processed of the associated operation is added (Fig. 2). An earliest starting time $h(S)$ is associated to each vertex S . $h(S)$ is the length of the shortest path from the origin to S .

The set of vertices V of the network constructed by TGA is composed by the three following types of vertices:

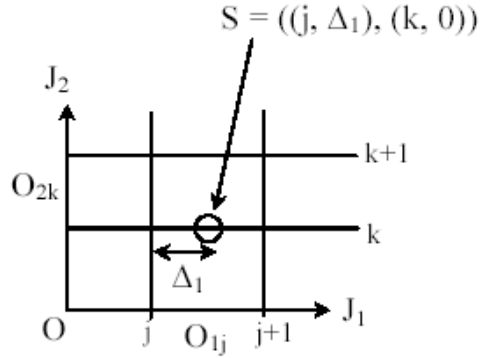


Figure 2. Vertex characterization

- Regular vertices, located at the intersection of horizontal and vertical lines, to which NW and SE corners of obstacles belong.
- Singular vertices, located on a horizontal (resp. vertical) line, which means that the execution of the operation of job J_2 (resp. J_1) has not started yet
- Waiting vertices, also located at the intersection of two lines, for which the execution of operations of jobs J_1 and J_2 has not started yet.

A singular vertex is created if the progression of only one job is possible (availability problem for the other job), whereas waiting vertices are created if the progression is possible for none of the two jobs. A waiting vertex is always a duplication of the regular vertex having the same geometric coordinates but not the same earliest starting time. The progression works as follow:

- If the operations of the two jobs cannot start at time $h(S)$, the earliest starting time of vertex S , a waiting vertex is then created.
- If there is an availability problem in the direction J_1 (resp. J_2), the progression is made along the vertical (resp. horizontal) line, that corresponds to the execution of the operations of job J_2 (resp. J_1) only, until job J_1 (resp. J_2) becomes available. A singular vertex, from which a diagonal progression is possible, is added as successor of S .
- If there is no availability problem, that is to say if the operations of the two jobs can be executed at time $h(S)$, the progression works as in the classical geometric approach.

3.2. AN EXTENDED APPROACH

We propose a generalization of TGA in order to deal with the flexibility property of the $J(MPM), NCwin \mid n = 2 \mid C_{max}$. Like the job shop problem, the scheduling of MPM job shop can be represented in the 2-dimensional plane with potential obstacles that depend on the assignment of machines for the two jobs. Let us define the vertices of the network, the successors of each vertex and the distance between any two vertices.

We develop the algorithm *SuccVertex* allowing to find the successors of each vertex $S = ((k_1, \Delta_{k_1}^1), (k_2, \Delta_{k_2}^2))$.

The algorithm progresses diagonally as long as it exists a couple of flexibility machines available for the operations of the two jobs. The corners *SE* and *NW* of the potential obstacles that could be reached from S are added as successors of S . If an availability problem occurs in one of the two direction the algorithm *SuccVertexAvailability* is used to define the successors of S in this case.

In fact the algorithm *SuccVertexAvailability* progresses horizontally or vertically until the progression in the two direction becomes possible, in this case a singular or a regular vertex is added as successor to S ; another availability problem occurs, and in this case a waiting vertex is added as successor of S or an unavoidable obstacle is hit.

The algorithm *SuccVertex* is in three steps.

Step 1 is an initialization step. Set Ph (resp Pv) is defined to keep the machines of job J_2 (resp J_1) allowing to progress. Set R is defined to keep the machines allowing to progress until meeting an horizontal and \ or a vertical. This set is used to progress in the next iteration of the program.

E_1 is the set of machine flexibility of O_{1k_1} and E_2 is the set of machine flexibility of O_{2k_2} . The initialization of these two sets depends on the type of the vertex, we have the following three cases:

- if S is a waiting vertex, E_1 and E_2 are made of machines used to create this vertex.
- if S is a singular vertex, E_1 and E_2 are made of machines used to create this vertex.
- if S is a regular vertex, $E_1 = \mu_{1v}$ and $E_2 = \mu_{2h}$, this means that all machines for O_{1v} et O_{2h} could be used for the progression.

In step 2, first we check the availability of the two machines for O_{1v} et O_{2h} . If these machines are available, we progress diagonally until a vertical and or an horizontal is met. In step3, we update the current time

$$J(MPM), NCwin | C_{max}$$

current_time, the sets E_1 , E_2 , v and h . The algorithm *SuccVertex* is stopped if the final obstacle is hit or when the diagonal progression is not possible because of availability problems or unavoidable obstacle.

AlgorithmSuccVertex

step1 $h = k_1, \Delta^1 = \Delta_{k_1}^1, v = k_2, \Delta^2 = \Delta_{k_2}^2, current_time = h(v)$,
 E_1 is the set of machine flexibility of O_{1v}
 E_2 is the set is the set of machine flexibility of O_{2h}
 $P_v = \emptyset, P_h = \emptyset, R = \emptyset$

step2
if $(h = n1 + 1)$ or $(v = n2 + 1)$ **then** add F as a successor. Stop
else
for all machine flexibility $t_1 \in E_1$ **do**
 for all machine flexibility $t_2 \in E_2$ **do**
 calculate the beginning $s_{current_time}^{t_1}, s_{current_time}^{t_2}$ and the end
 $t_{current_time}^{t_1}, t_{current_time}^{t_2}$ of the unavailability period of t_1 and t_2
 according to $current_time$
 check the availability of t_1 and t_2
 set $dir1 = 1$ if t_1 is available otherwise $dir1 = 0$
 set $dir2 = 1$ if t_2 is available otherwise $dir2 = 0$
 if $(t_1 = t_2)$ add corners SE and NW of potential obstacle as successors
 else
 if $dir1 = 1$ and $dir2 = 1$ **then**
 if $(p_{1v} - \Delta^1 < p_{2h} - \Delta^2)$ **then**
 $progress_vertic = 1$
 $P_v = P_v \cup t_1$
 $R = R \cup t_2$
 if $(p_{1v} - \Delta^1 > p_{2h} - \Delta^2)$ **then**
 $progress_horiz = 1$
 $P_h = P_h \cup t_2$
 $R = R \cup t_1$
 if $(p_{1v} - \Delta^1 = p_{2h} - \Delta^2)$ **then**
 $progress_diag = 1$
 $P_v = P_v \cup t_1$
 $P_h = P_h \cup t_2$
 if $dir1 = 1$ and $dir2 = 0$ **then**
 $progress_vertic_availability = 1$
 $P_{v-} availability = P_{v-} availability \cup t_1$
 $R_{h-} availability = R_{h-} availability \cup t_2$
 if $dir1 = 0$ and $dir2 = 1$ **then**
 $progress_horiz_availability = 1$
 $P_{h-} availability = P_{h-} availability \cup t_2$
 $R_{v-} availability = R_{v-} availability \cup t_1$
 if $dir1 = 0$ and $dir2 = 0$ **then**
 add the waiting place $(h, \Delta^1)(v, \Delta^2)$ as a successor,
 the earliest starting time of this vertex is fixed to the minimum between $T_{current_time}^{t_1}$
 and $T_{current_time}^{t_2}$

step 3**if**(*progress_vertic_availability* = 1) **SuccVertexAvailabilty** is called with $E_1 = P_v$ - availability, $E_2 = R_h$ - availability, *current_time*, h , v **if**(*progress_horiz_availability* = 1) **SuccVertexAvailabilty** is called $E_1 = R_v$ - availability, $E_2 = P_h$ - availability, *current_time*, h , v **if**(*progress_vertic* = 1) *current_time* = *current_time* + $p_{1v} - \Delta^1$ $\Delta^2 = \Delta^2 + p_{1v} - \Delta^1$ $v = v + 1$ $\Delta^1 = 0$ $E_1 = \mu_{1v}$ $E_2 = R$

goto step2

if (*progress_horiz* = 1) *current_time* = *current_time* + $p_{2h} - \Delta^2$ $\Delta^1 = \Delta^1 + p_{2h} - \Delta^2$ $h = h + 1$ $\Delta^2 = 0$ $E_2 = \mu_{2h}$ $E_1 = R$

goto step2

if(*progress_diag* = 1) *current_time* = *current_time* + $p_{1v} - \Delta^1$ $h = h + 1$ $\Delta^1 = 0$ $v = v + 1$ $\Delta^2 = 0$ $E_1 = \mu_{1v}$ $E_2 = \mu_{2v}$

goto step2

Algorithm SuccVertexAvailability

$h, v, current_time, E_1, E_2$ are given

step1

$\Delta^1 = 0, \Delta^2 = 0, Pv = \emptyset, Ph = \emptyset, R = \emptyset$

step2

if $(h = n1 + 1)$ or $(v = n2 + 1)$ **then** add F as a successor. Stop

else

for all machine flexibility $t_1 \in E_1$ **do**

for all machine flexibility $t_2 \in E_2$ **do**

 calculate the beginning $s_{current_time}^{t_1}, s_{current_time}^{t_2}$ and the end $t_{current_time}^{t_1}, t_{current_time}^{t_2}$ of the unavailability period of t_1 and t_2 according to $current_time$

 check the availability of t_1 and t_2

 set $dir1 = 1$ if t_1 is available otherwise $dir1 = 0$

 set $dir2 = 1$ if t_2 is available otherwise $dir2 = 0$

if $(t_1 = t_2)$ add corners SE and NW of potential obstacle as successors

else

if $dir1 = 1$ and $dir2 = 0$ **then**

if $current_time + p_{1v} - \Delta^1 < T_{current_time}^{t_2}$ **then**

$progress_vertic = 1$

$P_v = P_v \cup t_1$

$R = R \cup t_2$

if $current_time + p_{1v} - \Delta^1 > T_{current_time}^{t_2}$

 progression until $T_{current_time}^{t_2}$

 the singular vertex $(h, \delta^1 + T_{current_time}^{t_2} - current_time)(v, 0)$

 is added as successor of S

if $current_time + p_{1v} - \Delta^1 == T_{current_time}^{t_2}$

 progression until $T_{current_time}^{t_2}$

 the regular vertex $(h + 1, 0)(v, 0)$ is added as successor of S

if $dir1 = 0$ and $dir2 = 1$ **then**

if $current_time + p_{2v} - \Delta^2 < T_{current_time}^{t_1}$ **then**

$progress_horiz = 1$

$P_h = P_h \cup t_2$

$R = R \cup t_1$

if $current_time + p_{2v} - \Delta^2 > T_{current_time}^{t_1}$

 progression until $T_{current_time}^{t_1}$

 the singular vertex $(h, 0)(v, \delta^2 + T_{current_time}^{t_1} - current_time)$

 is added as successor of S

if $current_time + p_{2h} - \Delta^2 == T_{current_time}^{t_1}$

 progression until $T_{current_time}^{t_1}$

 the regular vertex $(h, 0)(v + 1, 0)$ is added as successor of S

if $dir1 = 0$ and $dir2 = 0$ **then**
 add the waiting place $(h, 0)(v, 0)$ as a successor,
 the earliest starting time of this vertex is fixed to the minimum between $T_{current_time}^{t_1}$
 and $T_{current_time}^{t_2}$
step 3
if ($progress_vertic = 1$)
 $current_time = current_time + p_{1h} - \Delta^1$
 $v = v + 1$
 $E_1 = \mu_{1v}$
 $E_2 = R$
 $\Delta^1 = 0$
 goto step2
if ($progress_horiz = 1$)
 $current_time = current_time + p_{2v} - \Delta^2$
 $h = h + 1$
 $\Delta^2 = 0$
 $E_2 = \mu_{2h}$
 $E_1 = R$
 goto step2

Distance Between Two Vertices S and S' :

Let us suppose without loss of generality that S and S' are located on horizontal lines, we have: $d(S, S') = \sum_{k_1}^{k_3-1} p_{1k} - \Delta^1 + \Delta^3$

For other cases, we need to check the availability of machines used to go from S to S' in order to calculate the distance between these two vertices, this is not a problem because for any arc going from vertex $S = ((k_1, \Delta^1), (k_2, \Delta^2))$ to vertex $S' = ((k_3, \Delta^3), (k_4, \Delta^4))$, machines of operation O_{1k} and O_{2j} are already assigned by algorithm *SuccVertex* for $k = k_1 \dots k_3, j = k_2 \dots k_4$.

Remarque: If F is a successor of S the distance between S and F is calculated using only available machines if possible or machines becoming available first. in fact we neglect the other paths.

Note that the distance could be calculated inside the algorithm *SuccVertex*.

THEOREM 1. *The set of vertices constructed by applying algorithm SuccVertex is sufficient to determine the optimal schedule.*

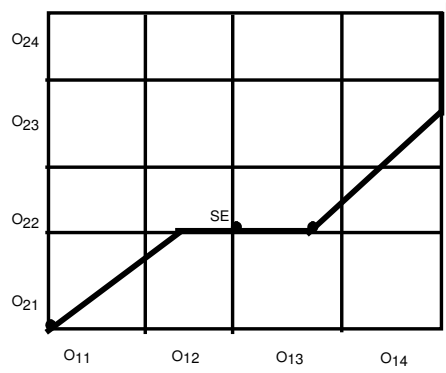


Figure 3. Shortest path

The theorem 1 is due to the fact that TGA gives the optimal schedule in the case of classical job shop (Aggoune, 2002).

3.2.1. Complexity Results

THEOREM 2. *The scheduling problem $J(MPM), NCwin \mid n = 2 \mid Cmax$ is polynomial and its complexity is at most equal to $O(As^4)$ where $m_1 = K_1 \times n_1, m_2 = K_2 \times n_2, K_i = \max(card(\mu_{ij}))$ and $s = \max(m_1, m_2)$, A is maximum number of unavailability periods that may occur on each machine.*

3.2.2. example

Let us consider the following example:

$$J_1 = \{M_3, M_5, 6\}, \{M_3, M_2, 5\}, \{M_1, M_5, 7\}, \{M_3, M_1, 6\}$$

$$J_2 = \{M_3, M_4, 7\}, \{M_2, 4\}, \{M_4, M_2, 6\}, \{M_5, M_3, 4\}$$

M_1 is supposed to be unavailable between times 16 and 22, M_2 between times 14 and 17, M_3 between times 8 and 13, M_4 between times 15 and 20, M_5 between times 20 and 25.

The shortest path is obtained in figure 3. It is composed of the following vertices:

$$((0, 0)(0, 0)); ((2, 0), (1, 0)); ((2, 6)(1, 0)); ((3, 0)(3, 0))$$

The optimal makespan is equal to 31, the assignment is as follow: O_{11} is assigned to M_5 , O_{21} to M_4 , O_{12} to M_2 , O_{22} to M_4 , O_{13} to M_5 , O_{23} to M_4 , O_{14} to M_3 , O_{24} to M_5 .

4. The general job shop problem with multi purpose machine and availability constraints

THEOREM 3. *A lower bound for $J(MPM), NCwin | C_{max}$ is calculated by considering all $O(N^2)$ pairs of jobs and taking the maximum solution values obtained by the polynomial algorithm of the previous section*

From the result of the previous section we can deduce a lower bound for the MPM job shop with limited machine availability (theorem 3) as well as a greedy heuristic to calculate a solution for the problem. This heuristic works as follows:

1. The two first jobs are optimally scheduled using the algorithm of the previous section
2. Additional unavailability periods, corresponding to the execution of operations of the two scheduled jobs, are fixed on each machine.
3. The algorithm is applied to the next two jobs of the sequence, taking into account the initial and the new unavailability periods. This procedure continues until all jobs are treated.

If the number of job is odd we need an insertion procedure to schedule the last job. It is based on the following rule: an operation of the last job is inserted in sort that it begins as early as possible, if we have the choice between two machines we choose the machine giving the smallest idle time.

4.1. EXAMPLE

Consider the following example with 5 jobs, 3 operations per job and 5 machines all data created aleatory (Table I).

M_1 is supposed to be unavailable between times 4 and 7, M_2 between times 7 and 10, M_3 between times 8 and 9, M_4 between times 5 and 8, M_5 between times 11 and 14.

We report in table II the makespan as well as the value of the lower bound. For this example the value of the lower bound is closed to the obtained makespan.

Note that the heuristic depend on the order of the input sequence. We are actually studying this major point in order to improve the global solution by controlling the initial input sequence.

Table I. Example : $N = 5$

J_1		J_2		J_3		J_4		J_5	
μ_{1j}	p_{1j}	μ_{2j}	p_{2j}	μ_{3j}	p_{3j}	μ_{4j}	p_{4j}	μ_{5j}	p_{5j}
5	3	4	5	2, 3	1	4	2	2, 3	5
4	4	4,3,2	2	5, 3,4	3	5	9	1	1
2,3	3	4	3	2, 3	4	2,3	3	2, 4,5	6

Table II. Example : $N = 5$

makespan	lower bound
26	24

5. Conclusion

We have investigated in this paper MPM job shop scheduling problems under availability constraints. We have proposed an extension of the temporized geometric approach to deal with the machine availability and the flexibility property of the problem. The developed algorithm is polynomial, as function of the number of operations , flexibilities and availability periods. The proposed approaches is used to develop a lower bound as well as a heuristic for the general problem. As future research, we will apply this method on some benchmarks with practical data and compare this method with a two-phase heuristic developed in a previous work (Zribi and al. , 2005).

References

Aggoune, R. (2002). Ordonnancement d'Ateliers sous Contraintes de Disponibilit des Machines. Ph.D. Thesis, Universit de Metz, France.

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

B. Jurisch (1992). Scheduling Jobs in Shops with Multi Purpose Machines. Ph.D Thesis, Universitt Osnabrck.

W. Kubiak, J. Blazewicz, P. Formanowicz, J. Breit, and G. Schmidt. Two-machine flow shops with limited machine availability. European Journal of Operational Research, 136: 528-540, 2002.

C. Y. Lee. Machine scheduling with an availability constraint. Journal of Global Optimization, 9: 395-416, 1996.

- C. Y. Lee. Minimizing the makespan in two-machine flowshop scheduling with availability constraint. *Operations research letters*, 20, 129-139, 1997.
- C. Y. Lee. Two-machine flowshop scheduling with availability constraints. *European Journal of Operational Research*, 114: 420-429, 1999.
- K.M. Lee, T. Yamakawa. A Genetic algorithm for general machine scheduling problems. *Int.Conf. on Conventional and knowledge-Based Electronics Systems*, Vol 2 pp60-66 Australia, 1998.
- G. Schmidt. Scheduling on semi identical processors. *Z. Oper.Res.*, A28, 153-162, 1984.
- G. Schmidt. Scheduling independent tasks with deadlines on semi-identical processors. *Journal of Operational research society*, 39, 271-277, 1988.
- Adiri, I., Bruno, J., Frostig, E., and Rinnooy Kan, A. H. G., Single machine flow-time with a single breakdown. *Acta Informatica*, 26, 679-696.
- Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, 121, 1-15.
- Akers, S. B. and J. Friedman (1955). A non-numerical approach to production scheduling problems. *Operations Research*, 3, 429-442.
- N.Zribi A. El Kamel P. Borne , Minimizing the Makespan for the MPM job-shop with availability constraints . submitted to International Conference on Industrial Engineering and Systems Management IESM 2005, May 16 - 19, Marrakech (Morocco)