7e Conférence Internationale de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 – Paris- France

« Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination.».

MINIMIZING THE NUMBER OF TARDY JOBS IN A PERMUTATION FLOWSHOP SCHEDULING PROBLEM WITH MINIMAL AND MAXIMAL TIME LAGS

E. DHOUIB, T. LOUKIL

Faculté des Sciences Economiques, Université de Sfax,
Route de l'Aérodrome, km 4, B.P. 1088 Sfax 3018,
Tunisie
emna_dhouib@yahoo.fr, taicir.loukil@fsegs.rnu.tn

J. TEGHEM

Service de Mathématique et de Recherche Opérationnelle (MATHRO), Faculté Polytechnique de Mons,
9, Rue de Houdain, 7000 Mons, Belgique
jacques.teghem@fpms.ac.be

Abstract: *In this paper, we consider permutation flowshop scheduling problems with minimal and maximal time lag constraints to minimize the number of tardy jobs. Time lags are defined as an interval of time that must exist between pairs s of successive operations of the jobs. Such constraints may be used to model various industrial situations. We first propose a mathematical programming formulation for the considered problem and we apply the mixed-integer programming subroutine of CPLEX to the proposed formulation. Then, seven heuristic algorithms and a simulated annealing algorithm (SAA) are proposed to solve the studied problem. Computational experiments to compare the proposed procedures are provided.*

KEYWORDS: *permutation flowshop, mathematical programming, heuristics, simulated annealing.*

1. INTRODUCTION

The permutation flowshop problem with minimal and maximal time lags can be defined as follows: a set of n jobs (j = 1, …, n) are to be processed on a set of m machines (k = 1, …, m) in the same order first on machine 1, second on machine 2, and so on until machine m. The processing time of job j on machine k noted as p_{kj} is fixed and known in advance. Each machine can process only one job at a time. Preemption is not allowed, so that once a job is started on a machine, it cannot be interrupted. For each job, minimal and maximal time lags between every pair of consecutive operations are required. When the waiting time between the end of the $(k-1)^{th}$ operation and the beginning of the k^{th} operation of the job j must be greater than or equal to a non negative value $\theta_{kj}^{\min}$, we talk about minimal time lag; when the waiting time between the end of the $(k-1)^{th}$ operation and the beginning of the k^{th} operation of the job j must be smaller than or equal to a non negative value $\theta_{kj}^{\max}$, we talk about maximal time lag. The waiting time between the end of the $(k-1)^{th}$ and the beginning of the k^{th} operation of the job j is then bounded by the minimal (lower bound) and the maximal (upper bound) time lags. The objective is to find a schedule of jobs such that the number of tardy jobs is minimized. Using the classical notation for scheduling problems, this problem can be noted $PFm/\theta_{kj}^{\min},\theta_{kj}^{\max}/\sum U_j$.

Several industrial situations may be modeled using time lag constraints:

- Minimal time lags may be used when waiting times between successive operations of the same job are imposed. Such constraints arise in some computer systems (Finta and Liu, 1994), in some automated medical laboratories (Chu and Proth, 1996), in the fabrication of printed circuits (Kim et al., 1996) and other fields.
- Maximal time lags may be used to model situations when the delay between operations must be not too long in order to avoid deterioration of products. Such constraint arises in the field of food industry biotechnologies and chemistry (Deppner, 2004) and agriculture (Foulds and Wilson, 2005).
- In some problems, minimal and maximal time lags are both required such as the problem of coupled tasks which arise in radar systems (Orman and Potts, 1997), the hoist scheduling problem (Manier and Bloch, 2003).

Yu (1996) studied the complexity of particular cases of the two-machine flowshop problem with minimal time lags. Dell'Amico (1996) considered the job shop and flowshop scheduling problems with two machines and minimal time lags. He gave complexity results for the preemptive and the non-preemptive cases and studied the relationship between these two problems. He also gave lower bounds and upper bounds for the flowshop problem and analyzed their worst case performances. Moreover, he defined a tabu search algorithm and proved the effectiveness of the proposed bounds through extensive computational results. Brucker et al. (2004) studied the flowshop and open shop problems with two machines involving minimal time lags and gave complexity results for various criteria. Yang and Chern

(1995) studied the two-machine permutation flowshop problem with maximal time lags to minimize the maximum completion time. They showed that the problem is NP-hard. They also proposed a branch and bound algorithm to solve the problem and provided computational results. Fondrevelle et al. (2006) derived new complexity results for some flowshop scheduling problems with minimal and maximal time lags and makespan objective. They developed an optimal branch and bound algorithm to solve the m-machine permutation flowshop problem with minimal and maximal time lags and carried out a performance analysis.

Although the flowshop scheduling problems have been extensively studied in the literature, few results exist for performance measures other than the makespan. Lawler and Moore (1968) studied the two-machine flowshop scheduling problem with common due date to minimize the number of tardy jobs. They developed a pseudo-polynomial dynamic programming algorithm with time complexity $O(nD^2)$ to solve the problem. Hariri and Potts (1989) developed lower bounds on the total number of tardy jobs and used these bounds in a branch and bound algorithm to optimally solve the flowshop scheduling problem minimizing the number of tardy jobs. Their computational results showed that problems involving more than 20 jobs and two machines are difficult to solve as many problems remain unsolved. Gupta and Hariri (1997) proposed four algorithms for four polynomially solvable special cases of the two-machine flowshop scheduling problem to minimize the number of tardy jobs. Koulamas (1998) investigated the complexity of two-machine flowshop problems with related due date criteria ($T_{\max}$, $\overline{T}$ and N_T). He showed that when certain restrictions are imposed on tnhe job processing times and their due dates, these problems are polynomially solvable while when these restrictions are not justified, these problems are NP-complete. Corce et al. (2000) proposed a branch and bound procedure for the two-machine flowshop scheduling problem to minimize the number of tardy jobs against a common due date. Xiang et al. (2000) studied the permutation flowshop scheduling problem with an increasing and decreasing series of dominating machines to minimize the number of tardy jobs. They presented a polynomial time solution algorithm to solve this problem. Bulfin and M'Hallah (2003) described an exact algorithm to solve the two-machine flowshop scheduling problem with weighted number of tardy jobs objective.

The rest of the paper is organized as follows: in section 2, we propose a mathematical programming formulation of the problem and some computational experiments. Seven heuristic algorithms are proposed in section 3. A simulated annealing algorithm is presented in section 4. In section 5, we provide computational results related to these different approaches. Finally, section 6 concludes the paper.

2. AN EXACT APPROACH

2.1. Mathematical programming formulation

In this section, we propose a mathematical programming formulation for the problem $PFm/\theta_{kj}^{\min},\theta_{kj}^{\max}/\sum U_j$. This problem is NP-hard. Suppose that a set of n jobs N = {1, 2, …, n} is available to be processed on a set of m machines M = {1, 2, …, m} at time zero. Let:

- d_j : the due date of the job j.
- C_{ik} : the completion time of the job in position i on machine k.
- x_{ij} : be one if the job j is scheduled in position i and not tardy, and zero otherwise.

A job is considered tardy if it is completed after its due date, otherwise, it is called an early job.
The problem is described by the following mixed integer linear programming formulation:

$$Max \quad Z = \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}$$

s.t.

$$\sum_{l=1}^{k}\sum_{j=1}^{n}\left[\left(p_{lj}+\theta_{lj}^{\min}\right)*x_{1j}\right] = C_{1k} \quad k=1,...,m \qquad (1)$$

$$C_{(i-1)k}+\sum_{j=1}^{n}\left(p_{kj}*x_{ij}\right)\le C_{ik}$$

$$i=2,...,n \ \text{and}\ k=1,...,m \qquad (2)$$

$$C_{i(k-1)}+\sum_{j=1}^{n}\left[\left(p_{kj}+\theta_{kj}^{\min}\right)*x_{ij}\right]\le C_{ik}$$

$$i=2,...,n \ \text{and}\ k=2,...,m \qquad (3)$$

$$C_{ik}\le C_{i(k-1)}+\sum_{j=1}^{n}\left[\left(p_{kj}+\theta_{kj}^{\max}\right)*x_{ij}\right]$$

$$i=2,...,n \ \text{and}\ k=2,...,m \qquad (4)$$

$$C_{im}\le\sum_{j=1}^{n}\left(d_j\ x_{ij}\right) \qquad i=1,...,n \qquad (5)$$

$$\sum_{j=1}^{n}x_{ij}\le 1 \qquad i=1,...,n \qquad (6)$$

$$\sum_{i=1}^{n}x_{ij}\le 1 \qquad j=1,...,n \qquad (7)$$

$$x_{ij}=\{0,1\} \qquad i=1,...,n \ \text{and}\ j=1,...,n \qquad (8)$$

$$C_{ik}\ge 0 \qquad i=1,...,n \ \text{and}\ k=1,...,m \qquad (9)$$

The objective of our problem is to minimize the number of tardy jobs which is equivalent to maximize the number of no tardy jobs. Constraints of type (1) equate the completion time of the job in position 1 on machine k to the sum of the processing times of the job in position 1 on the k first machines plus the sum of the minimum time lags between these operations. Constraints of type (2) state that the completion time of the job in position i on machine k is greater than or equal to the completion time of the job in position (i-1) on machine k plus the processing time of the job in position

i on machine k. Constraints of type (3) state that the completion time of the job in position i on machine k is greater than or equal to the completion time of the job in position i on machine (k-1) plus the minimal time lag between the (k-1)th operation and the kth operation of the job in position i plus its processing time on machine k. Constraints of type (4) state that the completion time of the job in position i on machine k is smaller than or equal to the completion time of the job in position i on machine (k-1) plus the maximal time lag between the (k-1)th operation and the kth operation of the job in position i plus its processing time on machine k. Constraints of type (5) insure that the completion time of the job in position i is not greater than its due date. Constraints of type (6) insure that at most one job is scheduled in the ith position, the inequality indicates that a position may be empty since not all jobs are early. Constraints of type (7) insure that each job is scheduled in at most one position, the inequality indicates that a tardy job is not assigned to any position. Constraints of type (8) force each x_{ij} to take the values 0 or 1 only. If the job j is scheduled in position i, x_{ij} takes the value 1 and this means that the job j is early. If the job j is not scheduled, x_{ij} takes the value 0 and this means that the job j is tardy. Constraints of type (9) force the completion time of the job in position i on machine k to be non-negative.

Several complexity results exist already in the literature concerned related problems. It is well known that problem $PFm//\sum U_j$ is NP-hard already for m = 2 and without any time lags (see Hariri and Potts (1989)). Fondrevelle et al. (2006) proved that problem $PFm/\theta_{kj}^{\min},\theta_{kj}^{\max}/\sum C_{\max}$ is NP-hard in the strong sense, even in the case of m = 2 and constant maximal time lags. Moreover, problems with similar structure are more difficult for the minimization of the number of tardy jobs than for the minimization of the makespan. So clearly, the problem treated in this paper, i.e. $PFm/\theta_{kj}^{\min},\theta_{kj}^{\max}/\sum U_j$ is NP-hard in the strong sense.

2.2. Computational experiments

We apply the mixed-integer programming subroutine of CPLEX to this mathematical programming formulation. To test the effectiveness of the CPLEX procedure, several test problems are randomly generated. A test problem is composed of the following inputs:

- processing times are generated from a uniform distribution [20, 100].
- the values of due dates are generated in the same way adopted by Hariri and Potts(1989). The objective of our problem is to schedule the jobs so that the number of jobs completed after their due dates is minimized. As indicated by Hariri and Potts, the hardness of this problem is likely to depend on the values of due dates. The more difficult problems correspond either to due dates not enough spread or located approximately in the centre of the total processing time range. In such situations, it is more difficult to determine which jobs will be early. Two parameters d^l and d^u are chosen to provide respectively lower and upper bounds on the relative values of due dates, with $d^l \in \{0.2; 0.4; 0.6\}$, $d^u \in \{0.4; 0.6; 0.8\}$ and $d^l < d^u$. We only tested the four cases (0.2, 0.4), (0.2, 0.8), (0.4, 0.6) and (0.6, 0.8). An estimate of the maximum completion time P is obtained by considering the machine which has the largest processing requirement $\left(\sum_{j=1}^{n} p_{k^*j} = \max_{1\le k\le m}\sum_{j=1}^{n} p_{kj}\right)$.

$$P=\left(\sum_{k=1}^{m}\sum_{j=1}^{n} p_{kj} + \sum_{j=1}^{n}(n-1)p_{k^*j}\right)\Big/ n$$

After fixing d^l and d^u, and calculating P, an integer due date for each job j is then generated from a uniform distribution $\left[Pd^l, Pd^u\right]$.

- the minimal and maximal time lags $\left(\theta_{kj}^{\min},\theta_{kj}^{\max}\right)$ are generated respectively from a uniform distribution $\left[0,\theta^{\min}\right]$ and $\left[0,\theta^{\max}\right]$.

To test the influence of the minimal and the maximal time lag constraints, four classes of problems were generated:

a) class $(0,+\infty)$: problems without time lag constraints, i.e $\theta_{kj}^{\min}=0$ and $\theta_{kj}^{\max}=+\infty$ $\forall$ k, j.

b) class $(8,+\infty)$: problems with minimal time lags only; the values $\theta_{kj}^{\min}$ are generated from a uniform distribution $[0,8]$ and $\theta_{kj}^{\max}=+\infty$ $\forall$ k, j.

c) class $(0,16)$: problems with maximal time lags only; $\theta_{kj}^{\min}=0$ and the values $\theta_{kj}^{\max}$ are generated from a uniform distribution $[0,16]$ $\forall$ k, j.

d) class $(8,16)$: problems with minimal and maximal time lags; the values $\theta_{kj}^{\min}$ and $\theta_{kj}^{\max}$ are generated from a uniform distribution $[0,8]$ and $[0,16]$ respectively $\forall$ k, j.

- the number of machines m $\in \{3; 5\}$.
- the number of jobs n $\in \{15; 20\}$.

For each problem, five instances were generated giving a total of 320 tests ($2\times2\times4\times4\times5$ for respectively the parameters n, m, $\left(d^l,d^u\right)$, $\left(\theta^{\min},\theta^{\max}\right)$ and the five instances). A time limit of 18000 seconds was set.

The following tables 1 and 2 provide the results of the tested problems. The first column refers to the size of the problem. The second and the third indicate respectively the two factors d^l and d^u, and the two parameters $\theta^{\min}$ and $\theta^{\max}$. Columns 4, 5, and 6 provide respectively the minimum, average and maximum CPU time (in seconds) required to solve a problem. Columns

7, 8 and 9 provide respectively the minimum, average and the maximum number of nodes of the branch and bound required to solve a problem, and, column 10 provides the number of unsolved problems UN within the time limit.

According to the obtained results, we can confirm that the influence of the values of due dates on the efficiency of the CPLEX procedure is very strong.

Indeed, the class of problems with d^l = 0.2 and d^u = 0.4 appears to be the easiest. All problems falling within this class were solved. Running time required to solve problems of this class is short compared with running time to solve the other classes of problems (The maximum running time required was 208,984 for a problem with 20 jobs and 3 machines).

(n, m)	(d^l, d^u)	$(\theta^{min}, \theta^{max})$	CPU Time			Number of nodes			UN
			Min	Avg	Max	Min	Avg	Max	
(15, 3)	(0.2, 0.4)	$(0, +\infty)$	0.313	1.297	1.859	0	854	1608	0
		(0, 16)	0.218	1.15	1.719	0	445.4	914	0
		$(8, +\infty)$	0.266	1.38	2.594	0	827.6	2402	0
		(8, 16)	0.218	1.287	2.328	0	588.8	1790	0
	(0.2, 0.8)	$(0, +\infty)$	1.266	8.09	14.406	400	7728.4	16275	0
		(0, 16)	1.906	69.775	273.813	205	65777	272753	0
		$(8, +\infty)$	2.219	10.26	30.234	852	8722	29422	0
		(8, 16)	1.328	20.11	64.359	328	15681	58398	0
	(0.4, 0.6)	$(0, +\infty)$	0.766	8.31	27.25	520	8098	25037	0
		(0, 16)	1.047	13.36	25.969	269	10202	20000	0
		$(8, +\infty)$	0.547	5.79	14.578	76	5244	13000	0
		(8, 16)	1.141	7.294	12.656	424	5187,8	11873	0
	(0.6, 0.8)	$(0, +\infty)$	0.344	23.32	66.172	19	14185	29823	0
		(0, 16)	9.188	1836.97	5444	1927	739254	1484701	0
		$(8, +\infty)$	0.859	115.8	350.203	229	131964	458000	0
		(8, 16)	3.719	145.975	384.359	2269	108158	310435	0
(15, 5)	(0.2, 0.4)	$(0, +\infty)$	0.891	1.15	1.328	96	205.4	519	0
		(0, 16)	0.859	1.387	1.75	64	120.6	209	0
		$(8, +\infty)$	0.75	1.047	1.438	69	184.4	354	0
		(8, 16)	1.25	1.612	2.25	116	205	298	0
	(0.2, 0.8)	$(0, +\infty)$	8.641	11.44	26.656	5957	7220	19643	0
		(0, 16)	15.969	55.83	134.781	6412	24303	50016	0
		$(8, +\infty)$	3.422	9.43	16.344	936	4725	9011	0
		(8, 16)	14.453	33.33	42.656	4245	9975	14949	0
	(0.4, 0.6)	$(0, +\infty)$	4.75	11.9	20.234	2355	7649	15340	0
		(0, 16)	9.109	21.753	52.719	3957	10189	12589	0
		$(8, +\infty)$	3.625	10.572	18.953	1383	5976	9119	0
		(8, 16)	22.734	34.628	50.656	8328	19154	32172	0
	(0.6, 0.8)	$(0, +\infty)$	2.469	33.34	58.531	1112	18297	30022	0
		(0, 16)	284.875	1770	4310.375	144431	787315	1890099	0
		$(8, +\infty)$	2.344	227.15	750.719	1135	129980	434371	0
		(8, 16)	129.594	730.78	1412	52533	321762	662417	0

Table 1. Mathematical programming approach for problems with 15 jobs and 3 machines and 15 jobs and 5 machines

(n, m)	(d^l, d^u)	$(\theta^{\min}, \theta^{\max})$	CPU Time			Number of nodes			UN
			Min	Avg	Max	Min	Avg	Max	
(20, 3)	(0.2, 0.4)	$(0, +\infty)$	2.563	18.9	66.734	1000	16373	64030	0
		(0, 16)	8.078	54.43	208.984	3063	25147	95055	0
		$(8, +\infty)$	3.156	10.58	21.969	1815	8493.6	23765	0
		(8, 16)	4.578	15.43	37.375	1524	6817.8	15509	0
	(0.2, 0.8)	$(0, +\infty)$	10.531	2903	14392	4180	2016477	10^7	0
		(0, 16)	110.422	1188.43	4888.9	39429	594926	1801000	0
		$(8, +\infty)$	11.406	464.328	1298.234	4968	97067	424549	0
		(8, 16)	330.781	845.41	5605.297	170460	889098	2303124	1
	(0.4, 0.6)	$(0, +\infty)$	0.328	4.125	15.203	0	2379	10404	0
		(0, 16)	20.31	1653	4205.54	6000	728936	1884745	0
		$(8, +\infty)$	1.25	3.42	10.906	276	1729.8	4789	0
		(8, 16)	29.281	4354	6511.172	11604	1444220	1594541	0
	(0.6, 0.8)	$(0, +\infty)$	0.406	1.475	2.203	0	527.2	1000	0
		(0, 16)	13.828	13.828	13.828	4267	4267	4267	4
		$(8, +\infty)$	0.422	2	4.219	4	958.8	3000	0
		(8, 16)	134.969	134.969	134.969	46236	46236	46236	4
(20, 5)	(0.2, 0.4)	$(0, +\infty)$	3	32.75	109.266	375	16600	61473	0
		(0, 16)	4.172	19.94	53.219	887	4900	14444	0
		$(8, +\infty)$	6.688	49.41	185.063	2245	19827	75706	0
		(8, 16)	4.984	30.31	55.391	943	10354	24374	0
	(0.2, 0.8)	$(0, +\infty)$	24.719	284.267	785.844	4754	94837	266341	1
		(0, 16)	736.359	1284.66	3645	155285	422282	1016507	1
		$(8, +\infty)$	79.922	1880	3279	20456	510707	1510574	1
		(8, 16)	740.75	3920	10235	168002	1303654	3476620	1
	(0.4, 0.6)	$(0, +\infty)$	131.469	1989.74	7279.828	39087	926024	3738630	0
		(0, 16)	827.375	3552.192	9489.266	232178	1190915	3016803	0
		$(8, +\infty)$	29.734	911.73	2304.7	9818	418650	1092375	0
		(8, 16)	154.2	1012.65	2442	130000	632487	1134975	2
	(0.6, 0.8)	$(0, +\infty)$	13.42	2144.87	6402	4818	502796	1498352	1
		(0, 16)	–	–	–	–	–	–	5
		$(8, +\infty)$	27.672	3400	7875.32	11084	1660357	3963455	1
		(8, 16)	–	–	–	–	–	–	5

Table 2. Mathematical programming approach for problems with 20 jobs and 3 machines and 20 jobs and 5 machines

This is not surprising because the due dates are small, so, only a few jobs can be executed before their due dates and the number of possible feasible schedules is small. The class of problems with d^l = 0.6 and d^u = 0.8 appears to be the hardest. Indeed, 20 of the 27 unsolved problems (UN) fall within this class. Running time required to solve a problem of this class is very high compared with running time required to solve a problem of the other classes. The classes with d^l = 0.2 and d^u = 0.8 and d^l = 0.4 and d^u = 0.6 are also difficult with each containing 5 and 2 unsolved problems (Table 3).

From the obtained results, we also observe that there is no significant difference between the running time

required to solve a problem without time lag constraints and a problem with minimal time lag constraints only. Contrary, the problems with maximal time lag constraints are more difficult to solve than those without time lag constraints or problems with minimal time lag constraints. Indeed, 2 of the 27 unsolved problems are problems without time lag constraints, 2 problems are with minimal time lags only, whereas 10 problems are with maximal time lag only and 13 problems are with minimal and maximal time lag constraints (Table 4). The reason is that without maximal time lags, the problem is more easily satisfied by scheduling the jobs later.

(d^l, d^u)	(0.2, 0.4)	(0.2, 0.8)	(0.4, 0.6)	(0.6, 0.8)	Total
UN	0	5	2	20	27

Table 3. Number of unsolved problems for each class of (d^l , d^u)

$(\theta^{min}, \theta^{max})$	$(0, +\infty)$	(0, 16)	(8, $+\infty$)	(8, 16)	Total
UN	2	10	2	13	27

Table 4. Number of unsolved problems for each class of ($\theta^{\min}$, $\theta^{\max}$)

Furthermore, the problem's size has an important influence on the behaviour of the CPLEX procedure. Indeed, the running time required to solve a problem grows with the number of jobs and machines. For the tested problems, 18 of the 27 unsolved problems are problems with 20 jobs and 5 machines (Table 5). This is not surprising, the problem size constitute a major difficulty for exact approaches. These approaches can only be applied for problems of small size and even in this case, running times required are very long.

(n, m)	(15, 3)	(15, 5)	(20, 3)	(20, 5)	Total
UN	0	0	9	18	27

Table 5. Number of unsolved problems for each class of (n, m)

3. HEURISTIC ALGORITHMS

In this section, we propose several heuristic algorithms to find approximately good solutions for the problem of $PFm/\theta_{kj}^{\min}, \theta_{kj}^{\max}/\sum U_j$ with $0 \le \theta_{kj}^{\min} \le \theta_{kj}^{\max} \le +\infty$. The proposed heuristic algorithms have the same structure. They are inspired from the algorithm of Moore (1968) to solve the $1//\sum U_j$ problem. They create a sequence π of the jobs according to a special selection rule R1. Jobs are sorted according to this sequence. When the first tardy job $\pi(i)$ is found, an early job u from the partial sequence is removed. The selection of the removed early job u results from the use of a simple rule R2. If the tardy job $\pi(i)$ becomes early, the removed job u is moved to a set of removed jobs T. Otherwise, the job $\pi(i)$ is moved to the set T and the removed job u is reinserted in its initial position. The algorithms stop when all jobs in the given sequence are sorted. The final sequence S is obtained by joining the set of the removed jobs T to the sequence of early jobs E. Thus, the proposed heuristics differ in the sequence π, obtained with rule R1, from which the algorithms start the search and/or in the selection rule R2 used to choose the early job to remove.

3.1. The general structure of the heuristic algorithms

The heuristic algorithms have the following general structure:

Step 1: Form a sequence $\pi = \{\pi(1), \pi(2), ..., \pi(n)\}$ of the n jobs according to a defined selection rule R1.
Step 2: Let: the set of removed jobs T = { ϕ }, the set of early jobs E = { ϕ }, the number of tardy jobs z = 0.
Step 3: Schedule the first job as soon as possible
a) Schedule the first job on the first machine

$C_{1,\pi(1)} = p_{1,\pi(1)}$

b) Schedule successively the other operations of the first job, using minimal time lags.
For k = 2 to m, do

$C_{k,\pi(1)} = C_{k-1,\pi(1)} + \theta_{k,\pi(1)}^{\min} + p_{k,\pi(1)}$

c) Verify if $\pi(1)$ is tardy or not

If $C_{m,\pi(1)} > d_{\pi(1)}$

Then T = T $\cup$ { $\pi(1)$ }

Else E = E $\cup$ { $\pi(1)$ }

End if
Step 4: Schedule the other jobs as soon as possible
For i = 2 to n, do
a) Schedule the first operation of the job in the ith position in the sequence on the first machine

$C_{1,\pi(i)} = C_{1,\pi(i-1)} + p_{1,\pi(i)}$

b) Schedule the other operations of the job in the ith position in the sequence with respect to the precedence constraints and minimal time lags
For k = 2 to m, do

$C_{k,\pi(i)} = \max\{C_{k,\pi(i-1)} + p_{k,\pi(i)}, C_{k-1,\pi(i)} + \theta_{k,\pi(i)}^{\min} + p_{k,\pi(i)}\}$

c) Make sure that the maximal time lag constraints are satisfied
For k = m to 2, do

If $C_{k,\pi(i)} > C_{k-1,\pi(i)} + p_{k,\pi(i)} + \theta_{k,\pi(i)}^{\max}$

Then $C_{k-1,\pi(i)} = C_{k,\pi(i)} - p_{k,\pi(i)} - \theta_{k,\pi(i)}^{\max}$

End if
End for
d) Verify if $\pi(i)$ is tardy or not

If $C_{m,\pi(i)} \le d_{\pi(i)}$

Then E = E $\cup$ { $\pi(i)$ }

Else
Find the job u in the partial sequence of early jobs E using a defined rule R2.
Remove u from the partial sequence E = E – {u}
Reschedule the jobs of the partial sequence
If $\pi(i)$ becomes early
Then T = T $\cup$ {u}, E = E $\cup$ { $\pi(i)$ }
Else
Reinsert u in the partial sequence E
T = T $\cup$ { $\pi(i)$ }
End if
End if
End for
Step 5:
The schedule is S = {ET}
For i = 1 to n, do
If $C_{m,S(i)} > d_{S(i)}$
Then z = z + 1
End if
End for
The number of tardy jobs is z.

To better understand how to verify if the maximal time lag constraints are satisfied (step 4.c.), we propose the following example: Set two jobs to schedule on three machines. The processing times, minimal time lags and maximal time lags are provided in Table 6. Considering the sequence σ = (1, 2). The jobs are first scheduled without verifying the maximal time lag constraints. Only the precedence constraints and the minimal time lag constraints are verified. Figure 1.a shows that for the first job, the maximal time lag constraints are satisfied, whereas for the second job, the maximal time lag constraints are violated. For this, the second and the first operations of the second job must be successively shifted (cf. Figure 1.b and Figure 1.c).

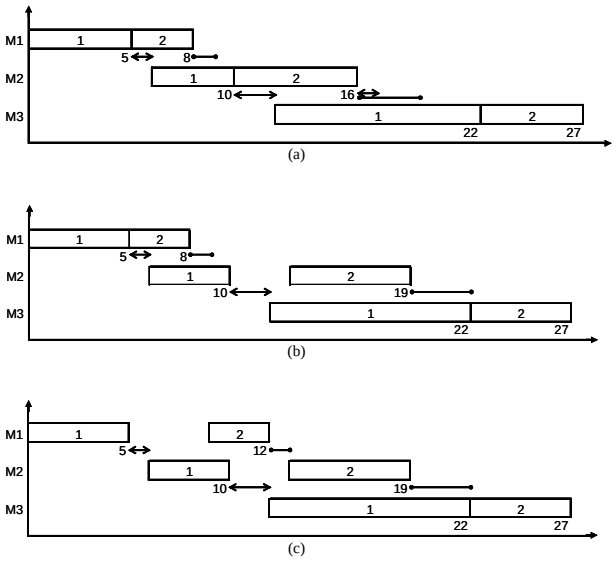

Figure 1. The scheduling of jobs

- (⟷): represents the minimal time lags.
- (●——●): represents the maximal time lags.

	p_{1j}	p_{2j}	p_{3j}	θ_{2j}^{min}	θ_{3j}^{min}	θ_{2j}^{max}	θ_{3j}^{max}
Job1	5	4	10	1	2	2	4
Job2	3	6	5	0	1	1	3

Table 6. Data of the example

Concerning rule R1, we propose four heuristics algorithms which start the search from a sequence of jobs according to the EDD rule and three heuristic algorithms which start the search from a sequence according to the SPT rule. They differ by the selection rule R2 used to remove early jobs when a tardy job is found.

3.2. Heuristics using the EDD rule

The rules R2 used are: remove the job with the largest sum of processing times (LPT), remove the job with the largest sum of processing times and minimal time lags (LPTMinTL), remove the job with the largest sum of processing times and maximal time lags (LPTMaxTL), remove the job with the largest sum of processing times and average of time lags (LPTAvgTL)

3.3. Heuristics using the SPT rule

The rules R2 used are: remove the job with the largest sum of minimal time lags (MinTL), remove the job with the largest sum of maximal time lags (MaxTL), remove the job with the largest sum of average of time lags (AvgTL).

4. SIMULATED ANNEALING ALGORITHM

A simulated annealing algorithm is also proposed to solve the problem. Our algorithm is composed of two principle ingredients as in Fortemps et al. (1996). The first, called scheduler, is a simple algorithm which has the role of scheduling the jobs according to a given sequence and evaluating the schedule. The second, called optimizer, is an implementation of simulated annealing. His role is essentially the search for a good solution by perturbing the current sequence of jobs and giving a new sequence.

4.1. The scheduler

For a given permutation $\pi = \{\pi(1), \pi(2), ..., \pi(n)\}$, the scheduler tries to find the optimal assignement of the n jobs in the given order on the m machines. The minimal and the maximal time lag constraints must be satisfied. After the scheduling of each job, the scheduler verifies if the job is tardy or not. If the job is tardy, the scheduler removes the job from the sequence of early jobs E and puts it in the sequence of tardy jobs T. The algorithm stops when all jobs are sorted and returns the value of the objective, the sequence of early jobs E and the sequence of tardy jobs T.

4.2. The optimizer

The role of the SA is to optimize the order of the jobs by perturbing a given sequence of jobs. At each iteration of the procedure, a new sequence is randomly chosen in the neighbourhood of the current sequence. For each new sequence, the scheduler is applied to obtain the value of our objective which is minimizing the number of tardy jobs.

5. COMPUTATIONAL RESULTS

The effectiveness of the proposed heuristics and the SAA was empirically tested. All algorithms were coded in visual C++ version 6.0 and the computational experiments were run on a HP Pentium IV. The inputs of the tested problems were generated in the same way adopted in Section 2 except for the minimal and the maximal time lag values because we only treat the most difficult situation of both constraints together (see section 2). The algorithms were applied to a set of problems with minimal and maximal time lags where: (n, m) $\in$ {(15, 3), (15, 5), (20, 3), (20, 5)}, $(d^l, d^u) \in$ {(0.2, 0.4), (0.2, 0.8), (0.4, 0.6), (0.6, 0.8)} ,and, $(\theta^{\min}, \theta^{\max}) \in$ {(8, 16), (40, 80) (100, 120)}. For these 48 different cases, 5 instances were generated, so that, the total number of tested problems is 240. To evaluate the effectiveness of the proposed algorithms, the generated problems were solved by the CPLEX procedure applied to the programming formulation presented in Section 2 to obtain, if possible, the optimal solutions. In the application of the CPLEX procedure, a limit time of 18000 seconds was set**.**

5.1. The effectiveness of the proposed heuristics

To compare the proposed heuristics, we compute the relative percentage deviation of each heuristic algorithm, in regard with CPLEX, δ_h (h = 1, 2, 3, 4, 5, 6, 7) as follows: $\delta_h = 100 \times (t_h - t_o) / (n - t_o)$, where:

- t_h : the total number of tardy jobs obtained by the heuristic.
- t_o : the total number of tardy jobs obtained by the application of the CPLEX procedure (optimal or best solution obtained).

So, for all instances -except for two instances in which a heuristic found a better solution than the one provided by CPLEX -, the values $(t_h - t_o)$, and thus δ_h, are non negative.

The following tables resumes the results. Table 7 indicates the number of times a heuristic found the best solution in regard with the 6 other heuristics. Table 8 indicates the number of times a heuristic found the solution provided by CPLEX, i.e, $(t_h - t_o) = \delta_h = 0$. And, table 9 gives the average percentage deviation δ_h of heuristic solution from the solution provided by CPLEX. In all these tables, we distinguish the three cases for $\left(\theta^{\min}, \theta^{\max}\right)$.

From the results given in tables 7, 8 and 9 below, it is clear that the heuristic algorithms which generate a schedule using the EDD rule (LPT, LPTMinTL, LPTMaxTL and LPTAvgTL) are more effective than those which generate a schedule using the SPT rule (MinTL, MaxTL and AvgTL). Indeed, heuristics using the SPT rule gave the best solution for less than the half of the tested problems, whereas the other heuristics gave the best solution in 68,75% to 81,25% of the tested problems. Furthermore, heuristics using the SPT rule gave the smallest number of solutions provided by CPLEX and the biggest average percentage deviation from this solution. This result is not surprising since we are attempting to minimize the number of tardy jobs. The due dates constitute a very important input in building a schedule. It is also seen from the results in the three tables that the heuristic using the EDD rule to create an initial schedule and the largest sum of processing times and minimal time lags (LPTMinTL) rule in the selection of the early job to remove, is the most effective. It gave the solution provided by CPLEX in 17,18% of the tests, the best solution among the 7 heuristics in 81,25% of the tests and with an average deviation from the solutions provided by CPLEX equal to 11,39%. The obtained results show also that the run time of the heuristic algorithms is very low (less than 1 second) compared to the run time required by the CPLEX procedure.

$(\theta^{\min}, \theta^{\max})$	EDD				SPT		
	LPT	LPTMinTL	LPTMaxTL	LPTAvgTL	MinTL	MaxTL	AvgTL
(100,120)	61	**65**	47	61	30	28	31
(40,80)	66	**68**	59	53	33	35	38
(8,16)	61	62	**63**	51	38	35	37
Total	188	**195**	169	165	101	98	106
Percent	78, 33	**81,25**	70,41	68,75	42,08	40,8	44,16

Table 7. Number of times a heuristic found the best solution in regard with the other heuristics

$(\theta^{min}, \theta^{max})$	EDD				SPT		
	LPT	LPTMinTL	LPTMaxTL	LPTAvgTL	MinTL	MaxTL	AvgTL
(100,120)	17	**18**	14	**18**	8	8	8
(40,80)	17	**18**	14	13	11	12	12
(8,16)	19	19	**20**	16	10	9	9
Total	53	**55**	48	47	29	29	29
Percent	16,56	**17,18**	15	14,68	9,06	9,06	9,06

Table 8. Number of times heuristic solution equals the solution provided by CPLEX

$(\theta^{min}, \theta^{max})$	EDD				SPT		
	LPT	LPTMinTL	LPTMaxTL	LPTAvgTL	MinTL	MaxTL	AvgTL
(100,120)	13,5	**13,04**	14,42	13,77	19,09	20,33	19,51
(40,80)	10,37	**9,89**	11,11	11,71	16,79	16,69	15,88
(8,16)	11,32	**11,26**	10,9	12,5	16,06	16,61	15,47
Overall	11,73	**11,39**	12,14	12,66	17,31	17,87	16,95

Table 9. Average percentage deviation of heuristic solution from the solution provided by CPLEX

5.2. The effectiveness of the proposed SAA

Taking into account the stochastic structure of the simulated annealing algorithm, preliminary tests are performed to determine the best control parameters that ensure the highest efficiency of the proposed algorithm in terms of the solution quality. Based on the results of the executed tests, we decided to use the following combination of control parameters: the initial sequence from which the procedure starts the search is randomly given, a neighbour sequence is selected by interchanging an early job with a late one, the initial sequence from which the procedure starts the search is randomly given, the initial temperature is equal to 20, a geometric cooling scheme is used with a cooling factor = 0.97, the maximum number of iterations "L" for each temperature value is equal to 2000, and finally, the search is terminated by the cooling mechanism, i.e, the temperature is near to zero (T = 0,01).

According to the obtained results, it is seen that the run time of the SAA is low (less than 1 second) and it increases slowly with the problem size, contrary to the CPLEX procedure which requires a huge computational effort which increases exponentially with the size of problems. This fact justifies the recourse to the metaheuristics to solve large instances. The results show also that the SAA found the best solution in 85% of tests with an average percentage deviation from the best value equal to 0.013 (Tables 10 and 11). Furthermore, for 30 unsolved instances to optimality by the CPLEX procedure within the fixed time limit, the SAA gave better solution for 10 instances, the same value for 18 instances and small worst value for 2 instances only. Taking into account the short run time and the simplicity of the SAA, these results may be quite satisfying.

Compared to the results given by the proposed heuristic algorithms, the SAA improved the best value found in 67% of the tested instances and in the 33% of the remaining instances, the performance is the same (Table 12). Since the $PFm/\theta_{kj}^{min}, \theta_{kj}^{max}/\sum U_j$ is NP-hard, the heuristics don't perform as well as SAA because they have more difficulties to escape the problem of local optima and the improvement is more important for large instances.

(n, m)	(15, 3)	(15, 5)	(20, 3)	(20, 5)	Total
Best value	55	52	48	49	204
Percent	0.91	0.86	0.8	0.81	0.85

Table 10. Number of times SAA solution equals the best solution

(n, m)	(15, 3)	(15, 5)	(20, 3)	(20, 5)	Overall
Avearge percentage deviation	0.008	0.016	0.014	0.014	0.013

Table 11. Average percentage deviation from the best solution

(n, m)	(15, 3)	(15, 5)	(20, 3)	(20, 5)	Total
Improvement	34	40	45	44	152
Percent	0.56	0.66	0.75	0.73	0.67

Table 12. Average percentage improvement of the best value found by the proposed heuristics

6. CONCLUSION

In this paper, we investigated the permutation flowshop problem with minimal and maximal time lags to minimize the number of tardy jobs $PFm/\theta_{kj}^{\min},\theta_{kj}^{\max}/\sum U_j$. This problem is NP-hard. We developed a mathematical programming formulation of the studied problem and we applied the mixed-integer programming subroutine of CPLEX to the proposed formulation. Computational experiments showed that the different values of the due dates and the introduction of the minimal and maximal time lag constraints have a big influence on the behavior of the CPU time. We proposed seven heuristic algorithms which have the same structure and differ in the rules used to construct a schedule. Four of these heuristics start the search from a sequence according to the earliest due date rule (EDD). The three other heuristics start the search from a sequence according to the shortest processing time rule (SPT). We also proposed a simulated annealing algorithm (SAA) to solve the problem. The results of computational experiments show that the heuristics using the EDD rule are more effective than those using the SPT rule. The application of the SAA gave results that are quite interesting in term of improvement of the solutions found by the proposed heuristics and quite satisfying in regard of the application of CPLEX, with a very small computing time.

Nevertheless, it will be interesting to compare the results of the SAA with these of other methods. For this reason we intend to go further with this problem, developing more sophisticated classical heuristics, inspired by the literature, an other metaheuristics like tabu search or genetic algorithms.

7. REFERENCES

Brucker, P., Knust, S., Cheng, T.C.E., Shakhlevich, N.V., 2004. Complexity results for flowshop and open shop scheduling problems with transportation delays. *Annals of Operations Research*, 129, p. 81-106.

Bulfin, R.L., M'Hallah, R., 2003. Minimizing the weighted number of tardy jobs on a two-machine flowshop. *Computers and Operations Research*, 30, p. 1887-1900.

Chu, C., Proth, J.-M., 1996. Single machine scheduling with chain structured precedence constraints and separation time windows. *IEEE Transactions on robotics and automation*, 12(6), p. 835-844.

Corce, F.D., Gupta, J.N.D., Tadei, R., 2000. Minimizing tardy jobs in a flowshop with common due date. *European Journal of Operational Research*, 120, p. 375-381.

Dell'Amico, M., 1996. Shop problems with two machines and time lags. *Operations Research*, 44(5), p. 777-787.

Deppner, F., 2004. Ordonnancement d'atelier avec contraintes temporelles entre opérations. *PhD Thesis*, Institut National Polytechnique de Lorraine, France.

Finta, L., Liu, Z., 1994. Scheduling of parallel programs in single-bus multiprocessor systems. *Rapport de Recherche INRIA*, 2302.

Fondrevelle, J., Oulamara, A., Portmann, M.-C., 2006. Permutation flowshop scheduling problems with maximal and minimal time lags. *Computers and Operations Research*, 33, p. 1540-1556.

Fortemps, Ph., Ost, Ch., Pirlot, M., Teghem, J., Tuyttens, D., 1996. Using metaheuristics for solving a production scheduling problem in a chemical firm: A case study. *International Journal of Production Economics*, 46-47, p. 13-26.

Foulds, L.R., Wilson, J.M., 2005. Scheduling operations for the harvesting of renewable resources. *Journal of Food Engineering*, 70, p. 281-292.

Gupta, J.N.D., Hariri, A.M.A., 1997. Two-machine flowshop to minimize the number of tardy jobs. *Journal of the Operational Research Society*, 48, p. 212-220.

Hariri, A.M.A., Potts, C.N, 1989. A branch and bound algorithm to minimize the number of late jobs in a permutation flowshop. European Journal of *Operational Research*, 38, p. 228-237.

Kim, Y.-D, Lim, H.G., Park, M.-W., 1996. Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process. European *Journal of Operational Research*, 91, p. 124-143.

Koulamas, C., 1998. On the complexity of two-machine flowshop problems with due date related objectives. *European Journal of Operational Research*, 106, p. 95-100

Lawler, E.L., Moore, J.M., 1968. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 15, p. 102-109.

Manier, M.-A., Bloch,C., 2003. A classification of hoist scheduling problems. *International Journal of Flexible Manufacturing Systems*, 15(1), p. 37-55.

Orman, A.J., Potts, C.N., 1997. On the complexity of coupled-task scheduling. *Discrete Applied Mathematics*, 72, p. 141-154.

Xiang, S., Tang, G., Cheng, T.C.E., 2000. Solvable cases of permutation flowshop scheduling with dominating machines. *International Journal of Production Economics*, 66, p. 53-57.

Yang, D.L., Chern, M.S., 1995. A two-machine flowshop sequencing problem with limited waiting time constraints. *Computers and Industrial Engineering*, 28(1), p. 63-70.

Yu, W., 1996. The two-machine flowshop problem with delays and the one-machine total tardiness problem. *Ph D Thesis*. Technishe Universiteit Eindhoven.