An algorithm for a bi-objective parallel machine problem with eligibility, release dates and delivery times of the jobs.

Manuel Mateo * **Jacques Teghem** ** **Daniel Tuyttens** ***

* *Universitat Politècnica de Catalunya(UPC), Avda.Diagonal 647,p7,08028 Barcelona,Spain (e-mail:manel.mateo@upc.edu).*
** *University of Mons, Polytechnic Faculty, Laboratory of Mathematics and Operational Research,Belgium (e-mail: jacques.teghem@umons.ac.be)*
*** *University of Mons, Polytechnic Faculty, Laboratory of Mathematics and Operational Research,Belgium (e-mail: daniel.tuyttens@umons.ac.be)*

Abstract: The scheduling of parallel machines is a well-known problem in many companies. Nevertheless, not always all the jobs can be manufactured in any machine and the eligibility appears. Based on a real-life problem, we present a model which has three different machines, called as high, medium and low level respectively. The set of jobs to be scheduled on these three parallel machines are also distributed among these three levels: one job from a level can be manufactured in a machine of the same or higher level. But a penalty appears when a job is manufactured in a machine different from the higher level. Besides, there are release dates and delivery times associated to each job. The tackled problem is bi-objective with the criteria: minimization of the final date - i.e. the maximum for all the jobs of their completion time plus the delivery time - and the minimization of the total penalty generated by the jobs. In a first step we revisited possible heuristics to minimize the final date on a single machine. In a second step a heuristic is proposed to approximate the set of efficient solutions and the Pareto front of the bi-objective problem. All the algorithms are experimented on various instances.

Keywords: scheduling; parallel machines; eligibility; release dates; delivery times; multi-objective optimization; Pareto front.

1. INTRODUCTION

The scheduling problem of parallel machines is very usual in the companies. A single operation must be done on a set of jobs and it is only necessary to perform each job in one of the available machine. In this work we deal with a particular problem within the scheduling of parallel machines, when some jobs cannot be done on any machine, what is known in literature as eligibility Leung and Li (2008).

Given a set of jobs ($j = 1, \dots, n$) to be scheduled on 3 parallel machines m_k, $k = 1, 2, 3$; these machines are different by their production quality called high ($k = 1$), medium ($k = 2$) and low ($k = 3$) level respectively. The same is done for the jobs: each of them is assigned to one of the 3 levels and a machine of level k can manufacture jobs of its own level and also of levels $k + 1$, 3. The processing time of a job is the same for any machine.

The characteristics of each job are the processing time p_j for the parallel machine operation, the release date r_j which can be a consequence of the previous operations received by the job, and the delivery time q_j as a result of the subsequent tasks or transport to the end of the production system.

Our problem is based on the scheduling process of a company devoted to paintings. It has a set of machines, which are reactors. The costs of manufacturing are important. The manager of the company prefers the use of the most modern resources, in our case the machine m_1. Nevertheless, if all the jobs were done in this machine, the final date of production would reach a very high value. In this case, the rest of machines would be completely free and available to manufacture. For this reason, some works from machine m_1 can be moved to the two others machines. But in such case, we define a penalty whose value is 1 if a job (of medium or low levels) is scheduled in machine m_2 and is 2 if a job (of low-level) is scheduled in the machine m_3. Therefore two objectives are taken into account when a feasible schedule is considered: to minimize the final date M of production, taking into account the delivery times, and to minimize the total penalty P due to the jobs scheduled on machines m_2 and m_3.

Section 2 indicates some references concerning related works. Section 3 presents the problem of scheduling n jobs with release dates and delivery times in a single machine, the heuristics proposed to treat this problem and some numerical experiments. After that, Section 4 focuses on the bi-objective problem, as the classification of machines, and some penalties are introduced. We propose a procedure based on an initial solution, and later a depth first search combined with a backtracking phase. Two examples illustrate how the algorithm is applied. Finally, Section 5 gives some conclusions and new perspectives to study this problem.

2. RELATED WORKS

The variety of studied scheduling problems is very large as described in several specialized books as those of Blazewicz et al. (2001) and Pinedo (2002) among many others. Despite our model has never been treated in the literature, a basic component of this model is the particular problem to schedule n jobs with release dates and delivery times on a single machine to minimize the final date. For this well-known problem which is $\mathcal{NP}$-Hard, see Garey and Johnson (1989), Schrage proposed a heuristic (see Blazewicz et al. (2001)) and Carlier (1982) an exact Branch-and-Bound using the Schrage's heuristic to generate an initial schedule. Later, the problem has been extended to parallel machines by Carlier (1987) and Gharbi and Haouari (2002). Some models of parallel machines with eligibility restrictions, which prohibit the schedule of some jobs to certain machines, have been analysed by Centeno and Armacost (1997), Centeno and Armacost (2004) and Leung and Li (2008). It is also the case of our model but in a different context with the particularity to consider a bi-objective minimization of the final date and a global penalty. So our original model takes place in the multicriteria scheduling theory as described by T'Kindt and Billaut (2002).

3. MINIMIZATION OF THE FINAL DATE OF A SINGLE MACHINE

3.1 The problem

We first consider the problem to schedule n jobs with processing times p_j, release dates r_j and delivery times q_j on a single machine to minimize the final date

$$M = \max_{j=1,\ldots,n} (C_j + q_j)$$

where C_j is the completion time of the job j.

As it will be necessary to solve very often this problem in the next section, we consider essentially heuristic methods.

3.2 The Schrage heuristic $H0$

This heuristic is based on the following. We note $\overline{U}$ the set of jobs not yet scheduled. Initially $\overline{U} = \{1, \ldots, n\}$ and the time $t = \min_{j\in\overline{U}} r_j$. Among the ready jobs $\{j \in \overline{U} \mid r_j \leq t\}$, the job i with the greatest q_i is scheduled at time t. Then $\overline{U} = \overline{U}\setminus\{i\}$ and $t = \max(t+p_i, min_{j\in\overline{U}}\, r_j)$ are updated before the next iteration.

3.3 Two other heuristics $H1a$ and $H1b$

Others simple heuristics are possible to determine the order of the jobs in the schedule.

Heuristic $H1a$: at each iteration determine the job $i \in \overline{U}$ corresponding to

$$\min_{j\in\overline{U}} (r_j, q_j) = \delta_i$$

If $\delta_i = r_i$, the job i is placed at the first place still available in the order. If $\delta_i = q_i$, the job i is placed at the latest place still available in the order. Then $\overline{U} = \overline{U}\setminus\{i\}$.

Heuristic $H1b$: at each iteration determine the job $i \in \overline{U}$ corresponding to

$$\max_{j\in\overline{U}} (r_j, q_j) = \delta_i$$

If $\delta_i = r_i$, the job i is placed at the latest place still available in the order. If $\delta_i = q_i$, the job i is placed at the first place still available in the order. Then $\overline{U} = \overline{U} \setminus \{i\}$.

For both heuristics in case of ex-aequo, place in front a job with the largest value q_i or with the smallest value r_i.

3.4 A different heuristic $H2$

This heuristic is based on the following dominance relation between jobs : $i \succ j$ if $r_i \leq r_j$ and $q_i \geq q_j$ with at least a strict inequality.

Clearly if $i \succ j$, i must be scheduled before j. At each iteration of $H2$, a non-dominated job is scheduled and the set D of non-dominated jobs inside $\overline{U}$ is updated. Inside D the jobs are ranked in increasing order of r_j and q_j . Let $l \in D$ be the largest index satisfying $r_l \leq t$. Only the jobs $\{j \in D \mid j \geq l\} = \tilde{D}$ are candidate to be scheduled. If there exist i and j, with $i < j$ such that $r_i + p_i \leq r_j$, the job j is not considered because it is more interesting to schedule job i. A pairwise comparison of the remaining jobs of $\tilde{D}$ based on the "lost" $r_n - r_m$ and the "gain" $q_n - q_m$, with $m < n$, is made.

If m^* and n^* are determined by

$$\min_{m,n\in\tilde{D}} (q_n - q_m) \;=\; q_{n^*} - q_{m^*},$$

$\tilde{D}$ is reduced in the following manner:

$$\tilde{D} = \tilde{D} \setminus \{m*\} \;\text{ if }\; r_{n^*} - r_{m^*} < q_{n^*} - q_{m^*}$$
$$\tilde{D} = \tilde{D} \setminus \{n^*\} \;\text{ if }\; r_{n^*} - r_{m^*} \geq q_{n^*} - q_{m^*}$$

till $|\tilde{D}| = 1$ and the remaining job is then scheduled.

3.5 An improvement algorithm

Based on the idea of the critical path inside the schedule proposed by Carlier (1982), an algorithm $H+$ is also proposed to improve the final date of an heuristic $H \in \{H0, H1a, H1b, H2\}$. We note this critical path $C = \{a, \ldots, p\}$ with thus a first job a and a last job p such that $M = C_p + q_p$. Such critical path begins at a time T_a and can be preceded by an empty time E_a of the machine: if $E_a \neq 0$ then $T_a = r_a$, if $E_a = 0$, then $T_a = r_a = 0$.

The main idea of $H+$ consists to analyse if the final date can be decreased either placing a job $j \in C$ with $r_j < r_a$ before the critical path C or placing a job $j \in C$ with $q_j < q_p$ after C.

For each of these two situations, we determine the sufficient conditions to be able to decrease the final date M. If these conditions are satisfied, the move of the job j is realized.

Conditions in the case $r_j < r_a$.

We note T_i the starting time of a job $i \in C$, $K = \{1, \ldots, j-1\}$ and $L = \{j+1, \ldots, p\}$. First of all, if $r'_j + p_j \leq T_a$ with $r'_j = \max(r_j, T_a - I_a)$, the job j can be removed of C and placed before T_a. Otherwise, the jobs of K will be delayed of the quantity $R_K = r'_j + p_j - T_a$ and a first condition to be able to decrease the final date M is

$$R_K \;<\; M - \max_{k \in K} (C_k + q_k).$$

The jobs of L can be advanced if it is compatible with their release date. So a second condition to be able to decrease the final date is

$$T_l \;>\; r_l \quad \forall\, l \in L.$$

Conditions in the case $q_j < q_p$.

First of all, there is no consequence for the jobs $k \in K$. As above, the jobs of L can be advanced if it is compatible with their release date. So a first condition to be able to decrease the final date is

$$T_l > r_l \quad \forall\, l \in L.$$

A second condition, related to the job j placed after C, is

$$q_j + I < q_p$$

with $I = \max\,(0, \max_{l \in L}\,(r_l - T_{l-1}))$ is the possible total idle time appearing inside the jobs of L. Finally, a third condition is related to the jobs $u \in U$ scheduled after the critical path C. If I_u is the total idle time appearing in the schedule after the job p but before the job u, this third condition is

$$\max\,(0, I - I_u) \;<\; M - (C_u + q_u) \quad \forall\, u \in U$$

3.6 Numerical experiments

A large set of 40 instances is randomly generated by the way proposed by Carlier (1982) and with $n \in \{10, 50, 100, 200, 500, 1000\}$.

Table 1 indicates the number of times the best value is obtained among the four algorithms $H0, H1a, H1b, H2$ and their respective improvement $H0+, H1a+, H1b+$ and $H2+$. The first line corresponds to 8 instances with $n = 10$ or 50; the second to 16 instances with $n = 100$ or 200; the third to 16 instances with $n = 500$ or 1000. The last line gives the results for the 40 instances.

Table 1. Results of the 8 heuristics H and $H+$ on 40 instances.

n	$H0/H0+$	$H1a/H1a+$	$H1b/H1b+$	$H2/H2+$
10, 50	5/8	7/8	0/8	7/8
100,200	9/16	16/16	0/16	13/16
500,1000	13/15	16/16	0/16	12/16
TOTAL	27/39	39/40	0/40	32/40

It appears that for these instances:

- The performance of $H1b$ is really bad;
- $H1a$ has the best performance followed by $H2$ and $H0$;
- $H+$ gives always the best value of the 40 instances, with only one exception with $H0+$ (see Section 5 for this exception).

4. THE BI-OBJECTIVE PROBLEM

Multicriteria scheduling problems are now classical studies (see T'Kindt and Billaut (2002)), in particular the determination of the Pareto front in the objective space. Our aim is here to approximate the Pareto front of the bi-objective problem described in the section 1.

We will use the following notations:

- m_k the machine of level k, $k = 1$, 2 and 3.
- J_k the subset of jobs of level k, $k = 1$, 2 and 3.
- I_k the subset of jobs assigned to machine m_k for a solution, with $I_2 \subseteq J_2 \cup J_3$ and $I_3 \subseteq J_3$.
- M_k the final date of the jobs I_k on machine m_k defined by $M_k = \max_{j \in I_k}\,(C_j + q_j)$.
- The two objectives to minimize are thus the global final date $M = \max_{k=1,2,3}\; M_k$ and the total penalty $P = |I_2| + 2|I_3|$.

Remark: To assign jobs to a machine, we always use the algorithm $H1a+$ of the section 2.

4.1 The initial solution with $P = 0$

This solution corresponds to the case $I_1 = \{1, \ldots, n\}, I_2 = I_3 = \emptyset$. All the jobs are thus scheduled on machine m_1 applying algorithm $H1a+$ and for this solution $M = M_1, M_2 = M_3 = 0$.

4.2 Depth first phase

Each iteration of this phase consists to generate a new solution with an increasing of one unit of the penalty P.

- If $M = M_1$, a job $j \in J_2 \cup J_3$ and belonging to the critical path of machine m_1 will be transferred from machine m_1 till machine m_2. For each possible j, the new value final date $M_1^{(j)}$ of machine m_1 with the jobs of $I_1 \setminus \{j\}$ and the new final date $M_2^{(j)}$ of machine m_2 with the jobs of $I_2 \cup \{j\}$ are measured and the job j^* to transfer is determined by $\min_j(\max(M_1^{(j)}, M_2^{(j)}))$. The new solution corresponds to $P + 1$ and $M = \max(M_1^{(j^*)}, M_2^{(j^*)}, M_3)$.
- If $M = M_2$, a job $j \in J_3$ and belonging to the critical path of m_2 will be transferred from machine m_2 till machine m_3. For each possible j, the new value $M_2^{(j)}$ of machine m_2 with the jobs of $I_2 \setminus \{j\}$ and the new final date $M_3^{(j)}$ of machine m_3 with the jobs of $I_3 \cup \{j\}$ are measured and the job j^* to transfer is determined by $\min_j(\max(M_2^{(j)}, M_3^{(j)}))$. The new solution corresponds to $P + 1$ and $M = \max(M_1, M_2^{(j*)}, M_3^{(j*)})$.

Nevertheless each time an iteration with $M = M_2$ succeeds to an iteration with $M = M_1$, this preceding iteration is registered with the corresponding j^* to be treated in the backtracking phase.

- If $M = M_3$, the backtracking phase of next section is applied.

4.3 The backtracking phase

We consider each registered solution, i.e. a solution with $M = M_1$ and such that the transfer of the job j^* from m_1 till m_2 generates a solution with $M = M_2$. This transfer of j^* produces the best solution with the value $P + 1$. Nevertheless it is possible that two different successive iterations - with at the first one the transfer of a job different than j^* - will produce a better solution with the value $P + 2$.

Effectively if another job $j_1 \neq j^*, j_1 \in J_2 \cup J_3$ is transferred from m_1 till m_2, we first obtain a less interesting solution with value $P + 1$, as $\max(M_1^{(j_1)}, M_2^{(j_1)}) > M = \max(M_1^{(j*)}, M_2^{(j*)})$.

But if $M_1^{(j_1)} > M_2^{(j_1)}$, a second iteration can be made to transfer another job $j_2 \neq j_1, j_2 \in J_2 \cup J_3$ from m_1 till m_2, or if $M_1^{(j_1)} \leq M_2^{(j_1)}$, a second iteration can be made to transfer a job $j_2 \in J_3$ from m_2 till m_3, and in each case it is possible that the new solution of value $P + 2$ is better that the preceding one obtained in the depth first phase, with the same value of the penalty.

So for each registered solution in the first phase, we check all the following situations consisting of two successive iterations:

- First transfer from m_1 till m_2 any job $j \neq j^*, j \in J_2 \cup J_3$ and belonging to the critical path of m_1;

- Then apply a second iteration, either if $M = M_1$ with the transfer of a job from m_1 till m_2 or if $M = M_2$ with the transfer of a job from m_2 to m_3 to obtain a solution with the value $P + 2$.

All these possibilities are checked and of course, if the best solution obtained is better that the solution obtained with the same value of the penalty in the first phase, we continue to apply the algorithm from this solution.

4.4 A first illustration

It concerns an instance with 20 jobs. The data of this instance are given in Table 2 with $J_1 = \{1, \ldots, 6\}, J_2 = \{7, \ldots, 16\}$ and $J_3 = \{17, \ldots, 20\}$. As the dimension of this instance is small, it is possible, by a long but complete enumeration, to obtain the exact Pareto front.

Table 3 presents in the first column the value P of the solutions, in the second column the value of M obtained with the heuristic described above and in the third column the value of M in the exact Pareto front. The symbol "*" indicates the solutions generated by the heuristic which are in fact dominated.

In Table 4, we describe each iteration of the heuristic indicating the final date (M_1, M_2, M_3) of each machine, the transfer from one machine to another and the complete solutions. Table 5 gives the solutions of the Pareto front dominating these obtained by the heuristic.

Analysing in details the structure of the exact efficient solutions, it appears that sometimes some jobs placed initially, at any preceding iteration, on the machine m_2 by the heuristic must in fact, in the solution with the next value of the penalty, be removed to the machine m_1 and replaced on machine m_2 by two others jobs. Clearly such possibility is not taken into account in the heuristic due to the very high combinatorial aspect of such mechanism. Nevertheless it gives a direction of research to improve the performance of the heuristic in the future.

4.5 A second illustration

It is also an instance with 20 jobs. The data of this instance are given in Table 6 with $J_1 = \{1, \ldots, 5\}, J_2 = \{6, \ldots, 11\}$ and $J_3 = \{12, \ldots, 20\}$.

For this instance, the heuristic obtains exactly the Pareto front which is presented in Table 7.

In Table 8, we describe each iteration of the heuristic indicating the final date (M_1, M_2, M_3) of each machine, the transfer of jobs from one machine to another and the complete solutions.

Table 2. Data of the first instance

j	r_j	p_j	q_j
1	1	8	4
2	10	6	1
3	19	9	18
4	16	2	9
5	6	4	7
6	8	4	16
7	19	2	9
8	6	2	7
9	16	1	5
10	2	9	8
11	20	6	3
12	14	7	2
13	16	4	8
14	9	9	10
15	12	9	13
16	3	10	5
17	1	7	3
18	12	10	11
19	8	10	12
20	17	3	11

Table 3. Results of the first instance

P	M (Heuristic)	M (Pareto front)
0	124	124
1	114	114
2	104	104
3	94	94
4	85	85
5	76	76
6	67	67
7	66 (*)	64
8	60	60
9	59	59
10	58 (*)	54
11	54 (*)	53
12	51	51
13	-	50
14	50 (*)	49

5. CONCLUSIONS AND PERSPECTIVES

As often, despite the simplicity of its formulation, the problem appears complex to solve. But some perspectives exist to improve the presented study and to extend the model treated.

- A first keypoint is clearly the problem of a single machine treated in section 2. If we analyze the behavior of the improvement algorithm $H+$, it appears that, even if this algorithm generates very often an optimal schedule, it is not always the case in its present form. Some new developments of $H+$ must thus be considered. In particular it seems that the main difficulty is coming in the case when the move of a job, before or after the critical path generates a new empty time E_a and thus modifies the subset of jobs forming this critical path : the order of the jobs inside this new critical path is not always the best one so that the corresponding final date is not the best one and the move can be rejected for this reason. For instance, it is what happens in the case mentioned in Table 1 where $H0+$ does not give the best value. Another question to analyze appears when a job j with $q_j < q_p$ is moved after the critical path. Presently in $H+$ this job j is placed immediately after the job p, but sometimes it appears more interesting to place this job j at another place after the job p.

- Concerning the bi-objective problem, as indicated in sub-section 4.4. the present heuristic does not allow to replace on machine m_1 (or m_2) a job placed at a preceding iteration on machine m_2 (or m_3). It is thus necessary to analyze if such possibility can be introduced inside the proposed method. A complete

Table 4. Iterations of the heuristic for the second instance

P	M_1, M_2, M_3	Transfer of jobs
0	124, 0, 0	all the jobs on m_1
1-17-10-16-5-8-6-19-14-15-3-18-20-4-7-13-9-11-12-2 // //		
1	114, 18, 0	job 16 from m_1 to m_2
1-17-10-5-8-6-19-14-15-3-18-20-4-7-13-9-11-12-2 // 16 //		
2	104, 33, 0	job 19 from m_1 to m_2
1-17-10-5-8-6-14-15-3-18-20-4-7-13-9-11-12-2 // 19-16 //		
3	94, 43, 0	job 18 from m_1 to m_2
1-17-10-5-8-6-14-15-3-20-4-7-13-9-11-12-2 // 19-18-16 //		
4	85, 46, 0	job 10 from m_1 to m_2
1-17-5-8-6-14-15-3-20-4-7-13-9-11-12-2 // 10-19-18-16 //		
5	76, 55, 0	job 15 from m_1 to m_2
1-17-5-8-6-14-3-20-4-7-13-9-11-12-2 // 10-19-15-18-16 //		
6	67, 64, 0	job 14 from m_1 to m_2
1-17-5-8-6-3-20-4-7-13-9-11-12-2 // 10-19-14-15-18-16 //		
7	66, 65, 0 (*)	job 9 from m_1 to m_2
1-17-5-8-6-3-20-4-7-13-11-12-2 // 10-19-14-15-18-9-16 //		
8	60, 58, 33	**backtracking** from $P = 6$ job 9 from m_2 to m_1, job 12 from m_1 to m_2, job 18 from m_2 to m_3
1-17-5-8-6-3-20-4-7-13-9-11-2 // 10-19-14-15-16-12 // 18		
9	59,59, 33	job 9 from m_1 to m_2
1-17-5-8-6-3-20-4-7-13-11-2 // 10-19-14-15-16-9-12 // 18		
10	58, 50, 39 (*)	**backtracking** from $P = 8$ job 9 from m_2 to m_1, job 8 from m_1 to m_2, job 19 from m_2 to m_3
1-17-5-6-3-20-4-7-13-9-11-2 // 10-8-14-15-16-12 // 19-18		
11	54, 54, 39 (*)	job 13 from m_1 to m_2
1-17-5-6-3-20-4-7-9-11-2 // 10-8-14-15-13-16-12 // 19-18		
12	51, 50, 39	**backtracking** from $P = 10$ job 13 from m_2 to m_1, job 17 from m_1 to m_2, job 17 from m_2 to m_3
1-5-6-3-20-4-7-13-9-11-2 // 10-8-14-15-16-12 // 17-19-18		
14	48, 50, 42 (*)	**backtracking** from $P = 12$ job 13 from m_2 to m_1, job 20 from m_1 to m_2, job 20 from m_2 to m_3
1-5-6-3-4-7-13-9-11-2 // 10-8-14-15-16-12 // 17-19-18-20		

Table 5. Solutions of the Pareto front

P	M_1, M_2, M_3
7	64, 64, 0
	1-17-16-5-8-6-3-20-4-7-13-9-2 // 10-19-14-15-18-11-12 //
10	54, 54, 39
	1-17-5-8-6-3-20-4-7-13-9-2 // 10-14-15-16-11-12 // 19-18
11	53, 48, 39
	1-17-5-8-6-3-20-4-7-13-9-11-2 // 10-14-15-16-12 // 17-19-18
13	50, 48, 42
	1-5-8-6-3-4-7-13-9-11-2 // 10-14-15-16-12 // 17-19-18-20
14	49, 49, 42
	1-5-8-6-14-3-4-7-9-2 // 10-16-15-13-11-12 // 17-19-18-20

Table 6. Data of the second instance

j	r_j	p_j	q_j
1	5	10	10
2	20	1	5
3	5	8	19
4	15	3	9
5	1	6	17
6	1	9	12
7	1	8	18
8	5	9	12
9	1	2	6
10	12	3	10
11	19	2	16
12	18	10	1
13	20	2	2
14	11	2	16
15	11	6	20
16	11	3	18
17	14	6	9
18	17	2	13
19	2	10	11
20	1	7	15

Table 7. Results of the second illustration

P	M (Heuristic and Pareto front)
0	111
1	101
2	92
3	83
4	75
5	68
6	61
7	58
8	55
9	53
10	50
11	49
12	47
13	46
14	44
15	43

different way to heuristically tackle the problem may be also to propose a particular multi-objective meta-heuristic adapted to the model.

- Finally it will be also interesting but difficult to extend the model.

    - A first direction is to modify the two considered penalties. Clearly fixing these two penalties to 1 and 2, when a job is scheduled on machine m_2 and m_3 respectively, makes the problem easier. With two any penalties P_1 and P_2 ($> P_1$), the problem becomes harder to solve.

    - A second direction is to introduce several parallel machines at each level k, modifying the model of section 3 inside the model studied by Gharbi and Haouari (2002). To examine if the heuristics in the present study are adaptable to such more complex model can be of course a next step of our work.

Table 8. Iterations of the heuristic for the second instance

P	M_1, M_2, M_3	Transfer of jobs
0	111, 0, 0	all the jobs on m_1
7-5-20-6-9-19-3-8-1-15-16-14-11-18-10-17-4-2-13-12 // //		
1	101, 23, 0	job 19 from m_1 to m_2
7-5-20-6-9-3-8-1-15-16-14-11-18-10-17-4-2-13-12 // 19 //		
2	92, 31, 0	job 6 from m_1 to m_2
7-5-20-9-3-8-1-15-16-14-11-18-10-17-4-2-13-12 // 6-19 //		
3	83, 40, 0	job 8 from m_1 to m_2
7-5-20-9-3-1-15-16-14-11-18-10-17-4-2-13-12 // 6-8-19 //		
4	75, 48, 0	job 7 from m_1 to m_2
5-20-9-3-1-15-16-14-11-18-10-17-4-2-13-12 // 7-6-8-19 //		
5	68, 48, 0	job 12 from m_1 to m_2
5-20-3-1-15-16-14-11-18-10-17-4-9-2-13 // 7-6-8-19-12 //		
6	61, 55, 0	job 20 from m_1 to m_2
5-3-1-15-16-14-11-18-10-17-4-9-2-13 // 7-20-6-8-19-12 //		
7	58, 58, 0	job 16 from m_1 to m_2
5-3-1-15-14-11-18-10-17-4-9-2-13 // 7-20-6-8-16-19-12 //		
8	55, 51, 23	**backtracking** from $P = 6$ job 16 from m_2 to m_1, job 17 from m_1 to m_2, job 19 from m_2 to m_3
5-3-1-15-16-14-11-18-10-4-9-2-13 // 7-20-6-8-17-12 // 19		
9	53,53, 23	job 18 from m_1 to m_2
5-3-15-16-14-11-1-10-4-9-2-13 // 7-20-6-8-18-17-12 // 19		
10	49, 50, 29	**backtracking** from $P = 8$ job 18 from m_2 to m_1, job 15 from m_1 to m_2, job 20 from m_2 to m_3
5-3-1-16-14-11-18-10-4-9-2-13 // 7-6-15-8-17-12 // 20-19		
11	49, 44, 38	job 15 from m_2 to m_3
5-3-1-16-14-11-18-10-4-9-2-13 // 7-6-8-17-12 // 20-15-19		
12	46, 47, 38	job 10 from m_1 to m_2
5-3-16-14-11-18-1-4-9-2-13 // 7-6-8-10-17-12 // 20-15-19		
13	46, 41, 40	job 17 from m_2 to m_3
5-3-16-14-11-18-1-4-9-2-13 // 7-6-8-10-12 // 19-15-20-17		
14	43, 44, 40	job 16 from m_1 to m_2
5-3-14-11-18-1-4-9-2-13 // 7-6-16-8-10-12 // 19-15-20-17		
15	43, 41, 43	job 16 from m_2 to m_3
5-3-14-11-18-1-4-9-2-13 // 7-6-8-10-12 // 19-15-16-20-17		

REFERENCES

Blazewicz J., Ecker K, H., Pesch E., Schmidt G., and Weglarz J. *Scheduling computer and manufacturing processes*. Springer-Verlag, Berlin Heidelberg New York, 2001.

Carlier J. The one machine sequencing problem. *European Journal of Operational Research*, 11, 42-47, 1982.

Carlier J. Scheduling jobs with release dates and tails on identical machines to minimize the makespan. *European Journal of Operational Research*, 29, 298-306, 1987.

Centeno G. and Armacost R.L. Parallel machine scheduling with release time and machine eligibility restrictions. *Computers and Industrial Engineering*, 33 (1-2), 273-276, 1997.

Centeno G. and Armacost R.L. Minimizing makespan on parallel machines with release time and machine eligibility restrictions. *International Journal of Production Research*, 42 (6), 1243-1256, 2004.

Garey M.R. and Johnson D.S. *Computers and intractability. A guide to the theory of NP-completeness*. Freeman Ed, 1989.

Gharbi R.L. and Haouari M. Minimizing makespan on parallel machines subject to release dates and delivery times. *Journal of scheduling*, 5, 329-355, 2002.

Leung J. Y. T. and Li C. L. Scheduling with processing set restrictions: a survey. *International Journal of Production Economics*, 116, 251-262, 2008.

Pinedo M.L. *Scheduling theory, algorithms, and systems*. Prentice-Hall, 2002.

T'Kindt V. and Billaut J.-Ch. *Multicriteria scheduling (theory, models and algorithms)*. Springer-Verlag, Berlin Heidelberg New York, 2002.