An algorithm for a biobjective parallel machine problem with eligibility and release and delivery times

Manuel Mateo, Jacques Teghem

9th International Conference of Modeling, Optimization and Simulation - MOSIM'12
June 06-08, 2012 – Bordeaux - France
"Performance, interoperability and safety for sustainable development"

AN ALGORITHM FOR A BIOBJECTIVE PARALLEL MACHINE PROBLEM WITH ELIGIBILITY AND RELEASE AND DELIVERY TIMES

Manuel MATEO

Dep. Organització Empreses / UPC
Av Diagonal, 647, 7th
08028 Barcelona - Spain
manel.mateo@upc.edu

Jacques TEGHEM

MATHRO / Faculté Polytechnique /UMons
9, rue de Houdain
7000 Mons - Belgique
jacques.teghem@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 situation, we present a company which has three different sets of machines, called as high-level, medium-level and low-level respectively. Besides, there are release times and delivery times related to the respective previous operations and following operations to the main process to be carried in the parallel machines. A set of n jobs to be scheduled on these m parallel machines are also distributed among levels. One job from a level can be manufactured in a machine of the same or higher level. Initially all the jobs are processed on the machines of high level. But a penalty appears when a job is manufactured in a machine different of this initial level. The proposed algorithm solves the problem with two criteria: the minimization of the completion time or makespan, C_{max}, and the minimization of the total penalty. The objective is to determine or to approximate the Pareto front. Several alternatives are presented according to different rules to move a job from one machine to another and compared through numerical experiments.*

KEYWORDS: *parallel machines, eligibility, release times, delivery times, 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 it in one of the available machines. The processing times of the operations are typically known (e.g. Pinedo, 2005). Some assigning and sequencing problems for two or more parallel machines have been described as highly complex (Blazewicz et al, 2001).

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 *n* jobs (j=1,…,n) to be scheduled on *m* parallel machines (i=1,…,m), these machines are distributed generally among *p* groups or levels (k=1,…,p). Particularly we will propose an algorithm for p=3, which means that machines and parts are separated into 3 levels (k=1 is considered for high-level machines; k=2 is considered for medium-level and k=3 for low-level). All the machines must be classified in one of the three levels. The same is done for the jobs: each of them is assigned to one of the levels. A machine in the level *k* can manufacture jobs of its own level *k* and also of lower levels. The processing time of a job is the same for any machine. This is known in the literature as nested processing set (Leung and Li, 2008).

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

Our problem is based in the scheduling of company devoted to industrial paintings. It has a set of machines, which are reactors. The costs of manufacturing are important. The managers of the company prefer the use of the most modern resources, in our case the high-level machines. Nevertheless, if all the jobs were done in this subset of machines, the makespan 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 machines of the high-level are moved to the other machines.

Moreover, machinery used for particular products, generally small products, is considered in the low-level. The medium-level machines can work these small products and also other products with longer dimensions. For this reason, the medium-level machines are also preferred to the low-level machines.

So, we define a penalty whose value is 1 if a job (of medium or low-levels) is scheduled in a medium-level machine and is 2 if a job (of low-level) is scheduled in a low-level machine. Penalties will be known as weights.

Therefore, two objectives are taken into account when a feasible schedule is determined: the minimum completion time or makespan, C_{max}, and the minimum total weight, W_{tot}.

A schedule will be said feasible if the following conditions are accomplished:

- Each machine processes at most one job at a time.
- A job is only processed in a single machine.
- Pre-emption is not allowed.
- Starting time is not lower than the release time: $t_j \geq r_j$.
- A job assigned to level *k* is processed in a machine of the same or a higher level.

Taking into account the release and delivery times of the jobs, our proposed algorithm refers to the algorithm described in Gharbi and Haouari (2002) if these two sets of times must be considered, but eligibility requires working with subsets of jobs at each level.

Section 2 presents the state of the art in parallel machines, with eligibility and release and delivery times, and some remarks on multicriteria. Section 3 describes the problem and the notation, while Section 4 introduces the multicriteria approach algorithm. Section 5 presents the computational experience. Finally, some conclusions are given in Section 6.

2 LITERATURE REVIEW

2.1 The parallel machine problem

According to the notation of Graham et al (1979), the problem can be noted as $Pm|r_j; q_j; M_j|(C_{max}, W_{tot})$, where job *j* can only be produced in a subset M_j of the *m* machines. A sub-problem can be solved for each value of the weight, i.e. the second criterion. Considering only the first criterion, the makespan, $Pm|r_j; q_j|C_{max}$ or $P|r_j; q_j|C_{max}$ is an extension of the classical identical parallel machine problem denoted by $P||C_{max}$, which is a basic problem in scheduling theory. However, we should remember that the preemptive version, denoted by $P|r_j, q_j, pmtn|C_{max}$, is solvable in polynomial time using a network flow (Horn, 1974). Considering only the makespan, the problem can be seen also as a generalization of the classical one machine problem $1|r_j; q_j|C_{max}$, which is known to be strongly NP-hard (Garey and Johnson, 1979). Therefore, the $Pm|r_j; q_j; M_j|(C_{max}, W_{tot})$ is NP-hard in the strong sense.

Parallel machine problems have received the attention from the research community since a long time. A survey of this literature is presented, for instance, in Pinedo (2002). If release and delivery times are added, the problem becomes $P|r_j; q_j|Cmax$, but despite its theoretical and practical interest, has received only a reduced attention. Indeed, the literature regarding this problem is relatively scarce.

To the best of our knowledge, the only exact algorithm in the literature for this problem was developed by Carlier (1987). Simple dispatching rules are analysed in Carlier (1987) and Gharbi and Haouari (2002). Bratley et al. (1975) developed a simple enumerative algorithm for the problem $P|r_j; d_j|C_{max}$, where d_j represents the deadline of job *j*. Lancia (2000) gives an enumerative method for the problem with two unrelated machines, denoted by $R2|r_j, q_j|C_{max}$.

Other similar problems have been investigated. On the one hand, there is the uniform parallel machine version, which was dealt by Dessouky (1998); he develops a branch-and-bound algorithm for $Q|r_j; q_j; p_j =1|C_{max}$. On the other hand, for the single machine problem $1|r_j; q_j|C_{max}$ some branch-and-bound algorithms have been proposed (Carlier, 1982).

Since $P||C_{max}$ is NP-hard in the strong sense, a problem with eligibility is strongly NP-hard as well. Following Leung and Li (2008), there are two important cases of eligibility that have received considerable attention in the literature: the nested processing set restrictions and the inclusive processing set restrictions. The first one has the property that for each pair M_j and M_k, either M_j and M_k are disjoint, or one of them is included in the other. The second one is a special case where the situation of disjoint M_j and M_k is not considered.

Our problem can be classified in the second group of situations, similar to what Hwang et al (2004) presented in the service industry. In this case of inclusive processing set restrictions, each M_j is associated with a single machine index a_j such that denotes the first or the last machine useful for the job *j*, depending on the sequence of machines. In our case, a_j means that a job *j* can be processed between machines 1 and a_j, once machines are sequenced by levels.

To the best of our knowledge, only Centeno and Armacost (1997) and once again Centeno and Armacost (2004) have worked on both characteristics at the same time. Nevertheless, they only look for the minimization of the makespan on parallel machines with release times and machine eligibility restrictions ($Pm|r_j; M_j|C_{max}$). They do not consider queue or delivery times.

2.2 The multicriteria problem

A survey of multicriteria research can be found in Ehrgott and Gandibleux (2002), more specifically on bicriteria scheduling in Nagar et al (1995). According to Loukil et al (2007), concerning multi-objective optimization problems we can distinguish five main approaches in the literature:

- Hierarchical approach: considered objectives are ranked in a priority order and optimized following this order.
- Utility approach: a utility function or weighting function, often a weighted linear combination of the objectives, is used to aggregate the considered objectives in a single one.
- Goal programming: all the objectives are taken into account as constraints which express some satisfying levels (or goals) and the objective is to find a solution whose values are as close as possible of the pre-defined goal for each objective.
- Simultaneous (or Pareto) approach: the aim is to generate, or to approximate in case of a heuristic method, the complete set of efficient solutions.
- Interactive approach: at each step of the procedure, the decision-maker expresses his preferences in regard to one (or several) solutions proposed. So, the method will progressively converge to a satisfying compromise between the considered objectives.

Our procedure can be classified into the fourth approach
We recall that for a multi-objective optimization problem:

$$\min_{X \in S} \; z_k(X) \quad k = 1,...,K \tag{1}$$

A solution $X^* \in \mathbf{S}$ is efficient or Pareto optimal (or non-dominated) if there is no other solution $X \in \mathbf{S}$ such that $z_k(X) \leq z_k(X^*)$ $\forall k$ with at least one strict inequality.

Despite their importance, scare attention has been given to multiple criteria scheduling problems, especially in multiple machine problems (T'Kindt and Billaut, 2002). This is due to the complexity of these combinatorial problems. From the analysis of the literature, often the methods proposed are either very complex to implement or only able to solve small size problems and with two objectives, or completely dependent on the model treated.

3 NOTATION

Let us consider the *m* machines divided in three levels. Instead of using k=1,2,3 for the different levels, we will denote by *h* the machines of the high level, *m* for the machines of the medium level, and *l* for the machines of the low level, respectively. The number of machines for each set of machines will be *mh*, *mm* and *ml*, respectively. In a similar way, the *n* jobs to be manufactured are also divided in similar groups. Then, the number of jobs will be *nh*, *nm* and *nl*, respectively. Finally, we thus have:

$$m = mh + mm + ml \tag{2}$$

$$n = nh + nm + nl \tag{3}$$

The *nl* jobs can be manufactured in machines of any level. The *nm* jobs of the second group can be treated in any of the *mm*+*mh* machines of medium and low levels. Finally, the *nh* jobs of the last group can be only manufactured in the *mh* machines of high level.

A job *j* (j=1,...,n) is defined by a processing time p_j; a release time r_j; a delivery time q_j and is classified into one of the three levels (h, m, l).

The job *j* will be scheduled in the machine between t_j, the starting time of job *j*, and t_j+p_j. Given t_j, the completion time of this job is computed as:

$$C_j = t_j + p_j + q_j \tag{5}$$

And obviously, the makespan is:

$$C_{max} = \max_j \{C_j\} \tag{6}$$

We also define $C_{max}H$, as the makespan only for the high-level machines; and equivalently $C_{max}M$ and $C_{max}L$ for the medium-level and low-level machines, respectively.

A solution will be represented by a vector σ, each one of its elements, σ_j, is composed of the assigned machine and the starting time of that job *j*.

On the other hand, weights are applied on the kind of level change:

$w_j = 1$ if *j* is assigned to one of the *mm* machines in σ

$w_j = 2$ if *j* is assigned to one of the *ml* machines in σ

and the total weight becomes:

$$W_{tot} = \Sigma_j \{w_j\} \tag{7}$$

The multicriteria objective is shown as:

$$\min_{X \in S} \; \{C_{max}(X), W_{tot}(X)\} \tag{8}$$

4 THE PROPOSED ALGORITHM

4.1 Some previous remarks on the algorithm

The algorithm is divided in two phases. It starts with a solution with null total weight, when all the jobs are assigned to high-level. I.e. any job (belonging to any level) is scheduled in one of the machines of high-level. This implies a first solution with the maximum C_{max} in the set of efficient solutions and the minimum weight, equal to 0.

While new efficient solutions can be found, an iteration of the second phase implies that one job is taken from an origin level machine and moved to a destination level machine.

Briefly, the substeps in this phase are:

- Select the origin level of the movement.
- Select a job to be changed to a different level (only a subset of jobs is available).
- Select the destination level of the movement.
- Machines in both levels are re-scheduled.
- The objectives of the new solution are evaluated (C_{max}; W_{tot}).

In our first proposal of the algorithm we establish a priority in the level changes:

- First, job changes between high-level and medium-level machines are tried.
- Then, job changes between high-level and low-level machines are tried.
- Finally, job changes between medium-level and low-level machines are tried.

We used different rules to select the job to be moved, among the subset of candidates (considering the origin and destination levels):

$j = \text{argminj}\ (r_j+p_j, q_j+p_j)$ (9)

$j = \text{argmaxj}\ (r_j+p_j, q_j+p_j)$ (10)

$j = \text{argminj}\ (r_j, q_j)$ (11)

$j = \text{argmaxj}\ (r_j, q_j)$ (12)

$j = \text{argminj}\ (p_j)$ (13)

$j = \text{argmaxj}\ (p_j)$ (14)

4.2 Heuristic 1 in case of one-machine per level

In order to schedule the jobs among the machines in a certain level, we use a procedure similar to what Garbi and Haouari (2002) proposed (see sub-section 4.3). Nevertheless, if there is only one machine per level a simpler procedure can be applied.

Let **J** a set of *n* jobs defined by the three times: r_i, p_i and q_i. We describe the algorithm with *n* jobs as in phase 1, but later this value can vary according to the distribution of jobs among levels.

For the initial sequence of jobs, two very similar heuristics are proposed: the first based on the minimum values, and the second on the maximum values. Their complexity is in O(nlogn)

Heuristic 1

Heuristic 1a

1. A job $s_0 \in J$ is such that r_{s_0} o q_{s_0} corresponds to min $\{r_j;q_j\}$ $\forall j=1,...,n$.
2. If s_0 has been chosen due its release time, the job occupies the first free position in the sequence. On the other hand, if s_0 has been chosen due its delivery time, the job occupies the last free position of the sequence.
3. $J = J \setminus \{s_0\}$. If $J = \varnothing$, go to Heuristic 1b; otherwise, go to step 1 in this Heuristic 1a.

Heuristic 1b

1. A job $s_0 \in J$ is such that r_{s_0} o q_{s_0} corresponds to max $\{r_j;q_j\}$ $\forall j=1,...,n$
2. Si s_0 has been chosen due its release time, the job occupies the last free position in the sequence. If s_0 has been chosen due its delivery time, the job occupies the first free position of the sequence.
3. $J = J \setminus \{s_0\}$. If $J = \varnothing$, stop; otherwise, go to step 1 in this Heuristic 1b.

At the end of both heuristics, a non-negative vector $\sigma = (t_1, t_2, ..., t_n)$ is obtained, where t_j is the starting time of job j ($t_j \geq r_j \forall j = 1;...;n$). For the next phases, the vector with a minimum makespan is taken.

Example 1

For an easier understanding, we will work with the low-level jobs in Table 1, where *k* is the level.

j	**j7**	**j8**	**j9**	**j10**	**j11**
k	1	1	1	1	1
$\mathbf{r_j}$	4	4	0	3	5
$\mathbf{p_j}$	8	6	5	4	2
$\mathbf{q_j}$	6	2	1	7	3

Table 1. Data for low-level jobs of Example 1.

There is a single low-level machine to produce the jobs. The heuristic is applied as follows:

Heuristic 1a

1. $J = \{7;8;9;10;11\}$; $s_0 = \{9\}$ and the minimum is r_s
2. Job 9 is assigned at the beginning of the sequence.

3. $J = \{7;8;10;11\}$ and return to step 1.

1. $s_0 = \{8\}$ and the minimum is q_s.

2. Job 8 is assigned at the end of the sequence.

And so on until $J = \varnothing$, the resultant vector is:

$$\sigma_1 = (9;10;7;11;8);\ C_{\max} = 27$$

Analogously, the heuristic 1b leads to vector:

$$\sigma_2 = (10;7;9;8;11);\ C_{\max} = 31$$

σ_1 is the preferred schedule.

4.3 Heuristic 2 in case of several machines per level

Gharbi and Haouari (2002) proposed an algorithm to minimize the makespan of a set **J** of *n* jobs to be performed on *m≥2* machines and results in two vectors $\sigma=(t_1,...,t_n)$ where t_i is the starting time for job *j* and $a=(a_1,\ldots,a_n)$ where a_j is the machine to which *j* is assigned.

But previously it is necessary to define two conditions to be evaluated. The jobs are sequenced by increasing order of release time $J_{r,m} = \{j_{(1)}, j_{(2)}, ..., j_{(m)}\}$. $J_{r.m}$ is a vector whose component $j_{(t)}$ is the job with the t-th arrival time and $j_0 \in J_{r,m}$ is such that $r_{j_0} + p_{j_0} = \min_{j \in J_{r,m}}(r_j + p_j)$. It is assumed that the Condition 1 (C1) is accomplished:

$$r_{j_0} + p_{j_0} \le r_{j_{(m+1)}} \qquad (15)$$

Then $t_{i0} = r_{i0}$ in an optimal schedule and the job j_0 can be removed from the set of jobs to be scheduled and placed in a set J_R. The problem can be considered symmetrical interchanging release and delivery times. An immediate consequence of this symmetry is the definition of $J_{q,m} = \{j_{(1)}, j_{(2)}, ..., j_{(m)}\}$, whose component $j_{(t)}$ is the job with the t-th delivery time in increasing order and $j_0 \in J_{q,m}$ is such that $q_{j_0} + p_{j_0} = \min_{j \in J_{q,m}}(q_j + p_j)$. In $J_{q,m}$ jobs are sequenced in increasing order of queue time. It is assumed that the Condition 2 (C2) is accomplished:

$$q_{j_0} + p_{j_0} \le q_{j_{(m+1)}} \qquad (16)$$

Similarly the job j_0 can be removed of the set of jobs to be scheduled and placed in a set J_Q. Then, the algorithm for scheduling of jobs in a selected level is:

0. Classification of jobs into $\overline{J}$, J_R and J_Q.

0.1. Initialize the three sets: $J_R = \varnothing$, $J_Q = \varnothing$, $\overline{J} = J$.

0.2. Check Condition C1.

0.2.1. If the number of jobs in $\overline{J}$ is no greater that m, i.e., if $|\overline{J}| \le m$, the algorithm stops.

0.2.2. If no job satisfies the Condition C1, go to 0.3; otherwise, $\overline{J} = \overline{J} \setminus \{j_0\}$; $J_R = J_R \cup \{j_0\}$ and go to 0.2.1.

0.3. Check Condition C2.

0.3.1. If the number of jobs in $\overline{J}$ is no greater that m, i.e. if $|\overline{J}| \le m$ the algorithm stops.

0.3.2 If no job satisfies the Condition C2, go to 0.4; otherwise, $\overline{J} = \overline{J} \setminus \{j_0\}$; $J_Q = J_Q \cup \{j_0\}$ and go to 0.3.1.

0.4. If no job is moved to a different set in 0.3, stop. Otherwise, go to 0.2.

1. The jobs in $\overline{J}$ are assigned to the machines to obtain a partial schedule σ. The first ready job is assigned to the first available machine.

2. Two new concepts are necessary:

- $u_i(\sigma)$ is the time in which the machine *i* is available given the schedule σ.
- σ^{-1} is the symmetric schedule of σ.

To integrate elements to the sets J_R and J_Q into σ, there are two parts: first, jobs of J_Q are assigned and then jobs of J_R. This step implies to apply the algorithm proposed by Gharbi and Haouari (2002).

2.1. All the jobs of the set J_Q are assigned.

2.1.1. If $J_Q = \varnothing$, go to 2.2.

2.1.2. Job $j_0 \in J_Q$ is such that $p_{j_0} + q_{j_0} = \max_{j \in J_Q}(p_j + q_j)$.

2.1.3. m_0 is called the machine with availability $u_0 = \min_{i=1,\ldots,n} u_i(\sigma)$.

2.1.4. Job j_0 is scheduled on machine m_0 and $u_0 = \max(u_0, r_{j_0}) + p_{j_0}$ is updated.

2.1.5. $J_Q = J_Q \setminus \{j_0\}$ and go to 2.1.1.

2.2. $\sigma = \sigma^{-1}$

2.3. All the jobs of the set J_R are assigned.

2.3.1. If $J_R = \varnothing$, go to 2.4.

2.3.2. Job $j_0 \in J_R$ is such that $r_{j_0} + p_{j_0} = \max_{j \in J_R}(r_j + p_j)$.

2.3.3. m_0 is called the machine with availability $u_0 = \min_{i=1,\ldots,n} u_i(\sigma)$.

2.3.4. Job j_0 is scheduled on machine m_0 and $u_0 = \max(u_0, q_{j_0}) + p_{j_0}$ is updated.

2.3.5. $J_Q = J_Q \setminus \{j_0\}$ and go to 2.3.1.

2.4. $\sigma = \sigma^{-1}$

In Step 2, jobs in the set J_O are first considered. The jobs are assigned in the non-increasing sequence of the addition of the delivery time and the processing time. At ach iteration, a job is assigned to the first available machine and the job is removed from J_O. Once these jobs are scheduled, the symmetric schedule is obtained in order to assign the jobs in set J_R, which will be allocated on the first available machine in the non-increasing sequence of ready plus processing times. As with the other set, each time a job is assigned, it is removed from the set J_R.

Example 2

We continue with similar data to the previous Example 1. Jobs and machines (*ml*=2) belong only to the low-level.

j	**j7**	**j8**	**j9**	**j10**	**j11**
k	1	1	1	1	1
r_j	6	4	0	3	8
p_j	8	5	2	1	4
q_j	6	2	1	4	7

Table 2. Data for low-level jobs of Example 2.

0.1. $J_R = \varnothing$, $J_Q = \varnothing$, $\bar{J} = J$.

0.2a. The set is $J_{r,m} = \{9,10\}$;

$\min_{j \in J_{r,m}} (r_j + p_j) = \min(2,4) = 2 \Rightarrow j_0 = \{9\}$.

Condition C1 ($r_{j_0} + p_{j_0} \le r_{j_{(m+1)}}$) is satisfied.

Consequently, the new set is $J_R = \{9\}$.

0.2b. Condition C1 leads to $J_{r,m} = \{10,8\}$.

$\min_{j \in J_{r,m}} (r_j + p_j) = \min(4,9) = 4 \Rightarrow j_0 = \{10\}$

As above, the Condition C1 is checked again so that the set is now $J_R = \{9;10\}$.

As above, the Condition C1 is checked again.

0.2c. Condition C1 leads to $J_{r,m} = \{8,7\}$.

$\min_{j \in J_{r,m}} (r_j + p_j) = \min(9,14) = 9 \Rightarrow j_0 = \{8\}$

As it is not accomplished, the algorithm goes to step 2.

0.3. Condition C2 leads to $J_{q,m} = \{7,8\}$

$\min_{j \in J_{r,m}} (p_j + q_j) = \min(7,14) = 7 \Rightarrow j_0 = \{8\}$

It is accomplished. Consequently, $J_Q = J_Q \cup \{7\}$, and the sets $J_R = \{9;10\}$; $\bar{J} = \{7;11\}$ and $J_Q = \{8\}$.

The algorithm finishes the classification, as $|\bar{J}| = m$.

1. Following with the example above, $\bar{J} = \{7;11\}$. Since the first job to arrive is job 7, it will be assigned to the first available low-level machine (*l1*). The following job, j11, is assigned to the other machine (*l2*). The vectors at the end of Step 1 are σ=(6,0,0,0,8) and a=(1,0,0,0,2), with C_{max}=20.

2.1.1. As $J_Q = \{8\}$, continue in 2.1.2.

2.1.2. As there is a single job, $j_0 = \{8\}$, then $p_{j_0} + q_{j_0} = 7$.

2.1.3. Given σ=(6,0,0,0,8), the availability time is $u_i(\sigma) = (14,12)$, whose minimum value is 12 and is associated to machine m_{l2}.

2.1.4. Job 8 is scheduled on machine m_{l2}; $u_0(m_{l2})$ = 12+5 = 17. Therefore: $u_i(\sigma)$=(14,17) and the new vectors are σ=(6,12,0,0,8), a = (1,2,0,0,2), with C_{max} = 20.

2.1.5. $J_Q = J_Q \setminus \{j_0\}$, but now $J_Q = \varnothing$ and finally go to 2.2.

2.2. The inverse σ= σ^{-1}=(6,3,0,0,8) is based on the subtraction of the ending processing time from C_{max} = 20.

2.3.1. As $J_R = \{9;10\}$, continue in 2.3.2.

2.3.2. Job $j_0 \in J_R$ is j10 as $r_{j_0} + p_{j_0} = \max_{j \in J_R}(r_j + p_j) = 4$.

2.3.3. Given $u_i(\sigma) = (14,12)$, $u_0 = 12$ comes from machine m_{l2}.

2.3.4. The job 10 is scheduled on machine m_{l2}: u_0 = 13. Therefore: $u_i(\sigma)$=(14,13) and σ= σ^{-1}=(6,3,0,12,8).

2.3.5. $J_Q = J_Q \setminus \{j_0\}$ and go to 2.3.1.

The step 2.3 is repeated to assign job 9, which is finally assigned to machine m_{l2}.

2.4. Finally, σ is inverted again: σ= σ^{-1}=(6,12,5,7,8), a = (1,2,2,2,2), with C_{max} = 20.

4.4 The main algorithm

Therefore, the algorithm is as follows:

Phase 0. Data input

Phase I: Assignment of all the jobs to high-level machines.

If *mh*=1, apply Heuristic 1 for all the jobs.
Otherwise, apply Heuristic 2 for all the jobs.
Compute C_{max}°

Phase II: Movement of jobs between different levels

II.1. Job movement from high to medium level
final = false
while (final = false)

```
    Search for job candidates
    if there is no candidate
        final = true
    else
        Search for a job between candidates according
        to the prefixed rule
        Reschedule jobs in high-level
        Reschedule jobs in medium-level
        Compute the new Cmax, CmaxH, CmaxM
        if Cmax < Cmax°
            Save the new solution
            Cmax° = Cmax
        else
            final = true
        endif
    endif
endwhile
```

II.2. Job movement from high to low level
This stage is analogous to II.1, but changing medium-level by low-level and $C_{max}M$ by $C_{max}L$.

II.3. Job movement from medium to low level
This stage is analogous to II.1, but changing high-level by medium-level and medium-level by low-level, and also $C_{max}H$ by $C_{max}M$ and $C_{max}M$ by $C_{max}L$.

5 COMPUTATIONAL EXPERIENCE

Computational experience is based on the implementation of the algorithm described in Section 4. The instances with $4 \leq m \leq 6$, n=20 and p=3 were generated similarly to what Gharbi and Haouari (2002) did; in fact they were created as in Carlier (1987). The number of jobs to produce is 20. For each number of jobs 100 instances are generated. The number of machines is 4, 5 and 6. Different distributions of machines are studied and are displayed in Table 3 and Table 4:

Distribution	4a	4b	4c	5a	5b	5c
mh	2	1	1	1	2	2
mm	1	2	1	2	1	2
ml	1	1	2	2	2	1

Table 3. Machine distribution in levels (m=4 and m=5).

Distribution	6a	6b	6c	6d	6e	6f	6g
mh	3	3	2	2	1	1	2
mm	2	1	3	1	3	2	2
ml	1	2	1	3	2	3	2

Table 4. Machine distribution in levels (m=6).

Processing times were generated using a discrete uniform distribution [1,10]. The proportion of high level jobs is between 20 and 30%; for the medium-level, 20-50%, and for the low-level, the rest, i.e. between 20 and 60%. The release and queue times are taken from a discrete uniform distribution [1,K(n/m)], where K is a positive integer equal to 3 or 5, as in Gharbi and Haouari (2002).

In the following tables, given the solutions in the Pareto front using two different rules A and B for the job selection from a origin level to another destination level, it is computed c(A,B) as the number of non-dominated solutions using the rule A by the solutions using rule B divided by the total number of solutions obtained using rule A. The respective rules A are in rows and rules B, in columns. The results for the overall 13 distributions are shown in Table 5.

	min (rp,qp)	max (rp,qp)	min (r,q)	max (r,q)	min (p)	max (p)
min(rp,qp)		0.26	0.42	0.37	0.90	0.22
max(rp,qp)	0.98		0.95	0.96	**1.00**	0.47
min(r,q)	0.96	0.34		0.61	0.97	0.25
max(r,q)	0.94	0.40	0.77		0.99	0.31
min(p)	0.54	0.28	0.40	0.34		0.24
max(p)	0.99	0.90	0.99	0.96	**1.00**	

Table 5: Proportion of non-dominated solutions (overall)

As Table 5 shows, the highest proportion is obtained with the rule which considers the largest processing time of a job in the candidate list. It is followed by the rule which considers the largest sum of processing and release or queue times, but clearly the first outperforms the second comparing both rules.

In the following tables (Table 6, Table 7 and Table 8), the results are shown according to the number of parallel machines *m*.

If we fix our attention to the max(p) rule in the three above tables, we can say this rule has a little better performance with m=4 than m=6, i.e. less machines, while the max(r+p,q+p) rule the performance for m=6 is slightly better than m=4.

	min (rp,qp)	max (rp,qp)	min (r,q)	max (r,q)	min (p)	max (p)
min(rp.qp)		0.23	0.41	0.33	0.92	0.19
max(rp.qp)	0.99		0.94	0.97	**1.00**	0.40
min(r.q)	0.96	0.33		0.58	0.96	0.24
max(r.q)	0.94	0.38	0.76		0.99	0.27
min(p)	0.46	0.22	0.37	0.27		0.19
max(p)	**1.00**	0.92	0.99	0.96	**1.00**	

Table 6: Proportion of non-dominated solutions (m=4)

	min (rp.qp)	max (rp.qp)	min (r.q)	max (r.q)	min (p)	max (p)
min(rp.qp)		0.27	0.42	0.39	0.91	0.22
max(rp.qp)	0.99		0.95	0.95	0.99	0.48
min(r.q)	0.97	0.35		0.63	0.98	0.25
max(r.q)	0.95	0.41	0.77		0.99	0.30
min(p)	0.55	0.28	0.39	0.36		0.23
max(p)	**1.00**	0.89	0.99	0.96	**1.00**	

Table 7: Proportion of non-dominated solutions (m=5)

	min (rp.qp)	max (rp.qp)	min (r.q)	max (r.q)	min (p)	max (p)
min(rp.qp)		0.28	0.43	0.39	0.88	0.24
max(rp.qp)	0.97		0.96	0.96	**1.00**	0.54
min(r.q)	0.94	0.35		0.63	0.98	0.27
max(r.q)	0.93	0.42	0.78		0.99	0.35
min(p)	0.61	0.33	0.45	0.40		0.29
max(p)	0.97	0.88	0.99	0.96	**1.00**	

Table 8: Proportion of non-dominated solutions (m=6)

Once we see that the max(p) and the max(r+p,q+p) rules are the most appropriate ways to select the job to be moved, we analyze it considering the different machine distributions.

Table 9 shows the comparison of solutions from max(p) rule against the rest of rules for a single and multiple machines.

According to Table 9, the highest number of non-dominated solutions is reached when there is a single machine in the high-level, while lowest number is obtained when there is a single medium-level machine and more than one high-level machines. We think the result for mh=1 is very reasonable, as there are more possibilities to decrease C_{max} and more efficient solutions can be obtained.

Finally, we must add that the computing times are very short, with values lower than a second.

	min (rp.qp)	max (rp.qp)	min (r.q)	max (r.q)	min (p)	All
mh=1	1.00	0.97	1.00	0.99	1.00	0.99
mm=1	0.97	0.87	0.98	0.94	1.00	0.95
ml=1	1.00	0.87	0.98	0.94	1.00	0.96
mh>1	0.98	0.85	0.98	0.93	1.00	0.95
mm>1	1.00	0.91	0.99	0.97	1.00	0.97
ml>1	0.98	0.91	0.99	0.97	1.00	0.97

Table 9: Proportion of non-dominated solutions for max(p) rule depending on the number of machines per level

6 CONCLUSIONS

The problem of parallel machines with eligibility and release and queue times has been studied, particularly when machines and jobs are divided into three levels. The studied multicriteria problem has two objectives to be minimized: the makespan and the total weight. This second value is positive if a job is not scheduled in the high-level machines. The set of solutions constitutes an approximation of the Pareto front.

As the initial situation is to produce all the jobs in machines of the high level, this solution shows a great makespan with null total weight. The rest of solutions in the Pareto front have a decreasing value in the makespan while the total weight simultaneously increases.

If we check different rules for the selection of the job to be moved from a level to another, the best results are achieved with the largest processing times, although the rule considering also release or delivery times added to the processing time has also a good performance.

Once have seen the results, we propose to continue the research with a dynamic combination of rules instead of using only one. We also intend to further tackle this problem using metaheuristics and determine accurate bounds.

ACKNOWLEDGMENTS

This work has been done with the support of grant DPI2007-61371 (Ministerio Educación y Ciencia, Spain).

REFERENCES

Blazewicz J., Ecker K.H., Pesch E., Schmidt G. and J. Weglarz, 2001. *Scheduling Computer and Manufacturing Processes*, Springer-Verlag.

Bratley P., Florian M. and P. Robillard, 1975. Scheduling with earliest start and due date constraints on multiple machines. Naval *Research Logistics Quarterly*, 22, p. 165 –173.

Carlier J., 1982. The One-Machine Sequencing Problem. *European Journal of Operational Research*, 11, p. 42–47.

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

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

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

Dessouky M.M., 1998. Scheduling identical jobs with unequal ready times on uniform parallel machines to minimize the maximum lateness. *Computers and Industrial Engineering*, 34, p. 793–806.

Ehrgott, M. and X. Gandibleux, 2002, *Multiobjective combinatorial optimization—theory, methodology and applications*, in: Ehrgott, M. and X. Gandibleux,

Multiple Criteria Optimization. State of the Art. Annotated Bibliographic. Surveys, Kluwer.

Garey, M.R. and D.S. Johnson, 1979. Strong NP-completeness results: Motivation, examples and implications. *Journal of the Association of Computer Machinery*, 25, p. 499–508.

Gharbi, A. and M. Haouari, 2002. Minimizing makespan on parallel machines subject to release dates and delivery times. *Journal of Scheduling*, 5, p. 329–355.

Graham R.L., Lawler E.L., Johnson D.S., Lenstra J.K. and Rinnooy K., 1979. Optimization and approximation in deterministic sequencing and scheduling theory: a survey. *Annals of Discrete Mathematics*, 5, p. 287-326.

Horn, W.A., 1974. Some simple scheduling algorithms. *Naval Research Logistics Quarterly*, 21, p. 177–185.

Hwang, H.-C., Chang, S.Y. and K. Lee, 2004. Parallel machine scheduling under a grade of service provision. *Computers and Operations Research*, 31 (12), p. 2055–2061.

Lancia, G., 2000. Scheduling jobs with release dates and tails on two unrelated parallel machines to minimize the makespan. *European Journal of Operational Research*, 120, p. 277–288.

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

Loukil, T., Teghem, J. and P. Fortemps, 2007. A multi-objective production scheduling case study solved by simulated annealing. *European Journal of Operational Research*, 179 (3), p. 709-722.

Nagar, A., Haddock, J. and S. Heragu, 1995. Multiple and bicriteria scheduling: A literature survey. *European Journal of Operational Research*, 81 (1), p. 88–104.

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

Pinedo M.L., 2005. *Planning and scheduling in manufacturing and services*, Springer.

T'Kindt, V. and J.-Ch. Billaut, 2002. *Multicriteria Scheduling (Theory, Models and Algorithms)*. Springer.