International Journal of Production Research, 2017
https://doi.org/10.1080/00207543.2017.1351634

A bi-objective parallel machine problem with eligibility, release dates and delivery times of the jobs

Manuel Mateo[a], Jacques Teghem[b] and Daniel Tuyttens[b]*

[a] *Universitat Politècnica de Catalunya (UPC), Barcelona, Spain;* [b] *Laboratory of Mathematics and Operational Research, Polytechnic Faculty, University of Mons, Belgium*

AQ1
AQ2

(*Received 14 November 2016; accepted 29 June 2017*)

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 m parallel machines with different level of quality from the highest level for the first machine till the lowest level for the last machine. The set of jobs to be scheduled on these m parallel machines are also distributed among these m levels: one job from a level can be manufactured in a machine of the same or higher level but a penalty, depending on the level, appears when a job is manufactured in a machine different from the highest level i.e. different from the first machine. Besides, there are release dates and delivery times associated to each job. The tackled problem is bi-objective with the criteria: minimisation of the final date – i.e. the maximum for all the jobs of their completion time plus the delivery time – and the minimisation of the total penalty generated by the jobs. In a first step, we analyse the sub-problem of minimisation of the final date on a single machine for jobs with release dates and delivery times. Four heuristics and an improvement algorithm are proposed and compared on didactic examples and on a large set of instances. In a second step an algorithm is proposed to approximate the set of efficient solutions and the Pareto front of the bi-objective problem. This algorithm contains two phases: the first is a depth search phase and the second is a backtracking phase. The procedure is illustrated in detail on an instance with 20 jobs and 3 machines. Then extensive numerical experiments are realised on two different sets of instances, with 20, 30 and 50 jobs, 3 or 4 machines and various values of penalties. Except for the case of 50 jobs, the results are compared with the exact Pareto front.

Keywords: scheduling; parallel machines; eligibility; release dates; delivery times; multi-objective optimisation

1. Introduction

The variety of studied scheduling problems is very large as described in several specialised books as those of Blazewicz et al. (2001) and Pinedo (2002) among many others. One of the reasons of this interest is that many real problems occurring in various industries are related to scheduling. We can distinguish many different scheduling situations: one machine, parallel machines, flow-shop, job-shop and open-shop. Moreover there may exist various different constraints to be satisfied by the schedule so that each case study is often specific and new with regard to the existing literature. It is the case for the particular problem considered in this paper.

AQ3
AQ4

Our problem is based on the scheduling process of a company devoted to painting. The jobs have a fixed processing time and are characterised by a release date which can be a consequence of the previous operations received by the job, and a delivery time as a result of the subsequent tasks or transport to the end of the production system. Preemption is not allowed. The workshop contains several machines $m_k,\ k = 1, \ldots, m$ which are reactors; for each job, the time spent in a machine is independent of the machine used but the quality of its production differs from each machine. The machine m_1 is the most recent and assures the best quality of product; the machine m_m is the oldest one and the other machines $m_k,\ k = 2, \ldots, m-1$ are intermediate machines with decreasing level of production quality. The manager of the company prefers of course the use of the most modern resources, i.e. the machine m_1. Nevertheless, if all the jobs were done on 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 can be moved from machine m_1 to other machines $m_k,\ k \geq 2$. Successive more restricted subset of jobs can be processed, respectively, on machines $m_2, \ldots, m_m$. But each time when a job is scheduled on the machine $m_k;\ k = 2, \ldots, m$, a penalty P_k is considered where P_k increases with the index k.

Therefore in our model two objectives are taken into account when a feasible schedule is considered: to minimise the final date M of production, taking into account the delivery times, and to minimise the total penalty P due to the jobs scheduled on machines $m_k,\ k = 2, \ldots, m$.

*Corresponding author. Email: daniel.tuyttens@umons.ac.be

We will use the following notations to formulate the above problem. A set of jobs ($j = 1, 2, \dots, n$) must be scheduled on m parallel machines $m_k,\ k = 1, \dots, m$; these machines are different by their production quality corresponding to the level k, from the highest level quality $k = 1$ till the worse one $k = m$. The same is done for the jobs and J_k is the subset of jobs of level k which can be assigned to machine m_1 till m_k. We have thus $J_m \subset J_{m-1} \subset \dots J_{k+1} \subset J_k \subset \dots J_1 = \{1, \dots, n\}$. A machine of level k can manufacture jobs of its own level and also of level superior to k.

Each job assigned to machine m_k with $k \geq 2$ generates a penalty fixed to P_k. The processing time p_j of the job j is the same for any machine; r_j and q_j are the release date and delivery time, respectively, of the job j.

Each feasible schedule is measured by two objectives to minimise. The first is M the final date of production of all the jobs, i.e. the value

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

where C_j is the completion time of the job j. The second is P the total penalty due to the jobs scheduled on the machines $m_k,\ k = 2, \dots, m$, i.e. the value

$$P = \sum_{k=2}^{m} P_k\, |I_k|$$

if I_k represents the set of jobs assigned to machine k.

Our aim is to determine, or to approximate, the Pareto front for the bi-objective problem (P, M). We recall here the notion of efficient schedule and of Pareto front. If $(P(S), M(S))$ is the performance of a feasible schedule S for the two objectives to minimise, S is an efficient schedule if there does not exist any other feasible schedule $\overline{S}$ such that $P(\overline{S}) \leq P(S)$ and $M(\overline{S}) \leq M(S)$ with at least a strict inequality. In the objective space, the set of solutions $\{(P(S), M(S)) \mid S \text{ is efficient}\}$ is call the Pareto front.

The rest of the paper is organised as follows. Section 2 presents the state of the art in parallel machines, with eligibility, release dates and delivery times, and some remarks on multi-objective optimisation. Section 3 analyses the problem of scheduling n jobs with release dates and delivery times in a single machine. This subproblem is particularly important for our study because it will be necessary to solve it many times in the proposed algorithm for the bi-objective problem described above. Four heuristics and an improvement procedure are compared, first on three didactic small instances, and after with numerical experiments on larger instances. Section 4 focuses on the general bi-objective problem. To approximate the Pareto front, we propose a procedure, based on an initial solution, combining a depth first search and a backtracking phase. The algorithm is illustrated in details on an instance of medium size so that the obtained results are analysed by comparison with the exact Pareto front generated by a complete enumeration of the feasible schedules. In Section 5, two different sets of 10 instances are generated for the numerical experiments. They are first treated with three machines and various values of penalties. Then these instances are extended to the case of four machines. Afterwards, instances with 30 jobs and 50 jobs are solved. Finally, some conclusions and perspectives are explained in Section 6. This article is a large extension of our conference paper Mateo, Teghem, and Tuyttens (2016), including a more general model, important developments of the methods and a very large set of numerical experiments.

2. Related works

According to the notation of Graham et al. (1979) and T'Kindt and Billaut (2002) our problem can be noted as $P/r_j, q_j, M_j/(C_{\max}, P)$, where M_j means eligibility i.e. the jobs can only be produced on a subset of the machines. Despite this model has never been treated in the literature, a basic component of this model is the particular problem $1/r_j, q_j/C_{\max}$ to schedule jobs with release dates and delivery times on a single machine to minimise 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. This problem will be further investigated in Section 3 because it will be extensively treated inside the bi-objective algorithm, of Section 4. Later, it has been extended to the model $P/r_j, q_j/C_{\max}$ by Carlier (1987), Gharbi and Haouari (2002, 2007). However, we should remember that the pre-emptive version, denoted by $P/r_j, q_j, pmtn/C_{\max}$, is solvable in polynomial time using a network flow (see Horn 1974). AQ5

The models of parallel machines which prohibit the schedule of some jobs to certain machines are known as problems with eligibility restrictions. Such problems have been analysed by Centeno and Armacost (1997, 2004) and Leung and Li (2008). Liao and Sheen (2008), not only combine eligibility constraints and release times with the objective of minimising the maximum makespan, but also add arbitrary pattern of machine availability intervals. A similar work with only two levels and without release times is presented in Lin and Liao (2008).

According to Lee, Leung, and Pinedo (2011), 'jobs with release dates and eligible sets have seldom been studied in the literature. A few experimental results can be found in some papers'. They consider the problem with equal processing times and release dates. For the same problem, Shabtay, Karhi, and Oron (2015) present an algorithm allowing also the rejection of jobs.

Recently, Leung and Li (2016) provide an expository update of this kind of problems, as according to them there has been a significant increase interest. They define some kinds of eligibility. Our case can be included in the inclusive processing set restriction as the subset J_k of machines for jobs of level k is included in the subset $J_{k'}$ (with $k' < k$). Li and Li (2015), Li and Lee (2016) and Li (2016) have presented some works, but always with a single objective. The two first ones solve the problem with equal processing times for all the jobs. And the last one considers queue time, but all the jobs are available at time 0. AQ6

Sometimes, machine eligibility constraints appear in a hybrid flowshop environment (see Wang, Yang, and Chang 2012) where makespan is minimised, while considering parallel batch, release time, and machine eligibility, but queue times are omitted. Other times, an order is composed of several product types, each product type can be produced on one and only one machine which is dedicated to that product type, as in Leung, Li and Pinedo (2008), where there are also release times. AQ7

Recalde et al. (2010) face the problem of minimising the makespan on restricted related parallel machines (another name for eligibility), and use different neighbourhoods, namely jump, swap, push and lexjump (lexicographical jump), to obtain new solutions.

One particular case of eligibility is the grade of service (GoS), which divides the customers in levels and the assignment takes this into account. According to Tseng et al. (2017), 'most studies on GoS eligibility focus on parallel machine scheduling with makespan objective'.

It is stated that $P/r_j, q_j/C_{\max}$ is equivalent to maximum lateness problem $P/r_j, d_j/L_{\max}$, where d_j is the due date of job j. Haouari and Gharbi (2003) analyse this last problem and give a semi-preemptive solution. Ying and Cheng (2010) and Lin et al. (2011) return on this problem, but also considering set-up times.

It is also the case of our model but in a different context with the particularity to consider a bi-objective minimisation of the final date and a global penalty. Such bi-objective problem is a particular case of the large field of multi-objective combinatorial optimisation (see Ehrgott and Gandibleux 2002). Despite their importance, reduced attention has been given to multi-objective scheduling problems, especially in multiple machine problems (see T'Kindt and Billaut 2002). Lei (2009) provides a review of the literature on the scheduling problems, in which the number of works for a parallel machine environment is small. Moreover, in this category of models, often the objectives are aggregated in a single function so that a unique optimal solution is determined. It is not the case in the present paper, as the aim is to determine the set of efficient solutions for the two objectives. Mohri, Masuda, and Ishii (1999) consider release times, due dates, set-up times in a bicriteria problem, in which the objectives are the minimisation of the makespan and the maximum lateness. The works considering only two criteria for identical parallel machines follow basically two lines of research: ones deal with an objective for production and another for maintenance (Berrichi and Yalaoui 2013; Moradi and Zandieh 2010); the others take into account the combination of the total completion time or makespan and the tardiness (Zarandi and Kayvanfar 2015). On the other hand, in case of unrelated parallel machines, there is an important number of articles (for instance Naderi-Beni et al. 2014). AQ8

Generally, the objectives are classical scheduling criteria (makespan, average completion times, tardiness,...) and only very few studies, concerning particular contexts of applications, consider job assignment costs, like set-up costs or tool changing costs where two successive jobs belong to two different classes (Chapter 4.4 of T'Kindt and Billaut 2002), rejection costs when it is possible to decide to not schedule some jobs (Moghaddam, Yalaoui, and Amodeo 2015), or penalty costs like in our model, when exist different levels of production.

3. Minimisation 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 minimise the final date $M = \max_{j=1,\dots,n} (C_j + q_j)$ 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 only consider heuristic methods. As far as we know, in the literature there exists only one heuristic due to Schrage (Blazewicz et al. 2001). We will note H_0 this heuristic. In addition, we propose three other heuristics called H_{1a}, H_{1b} and H_2 (Section 3.2). The pseudo-code of these methods is described in that section.

In a first step, these algorithms will be illustrated on three didactic examples (Section 3.3). The first is identical as the one considered in Carlier (1982). These illustrations will allow to compare the mechanisms of these four heuristics and to analyse their respective characteristics.

Then, we also propose an improvement algorithm called H^+ (Section 3.4) which can be applied to the result of any heuristic H. We describe its pseudo-code and prove its interest on the three didactic instances. AQ9

In a second step, numerical experiments on large instances will be given in Section 3.5.

3.2 *Four heuristics*

3.2.1 *The Schrage heuristic H_0*

The pseudo-code of the Schrage heuristic H_0 is shown in Algorithm 1. This heuristic is based on the following principle and builds the schedule progressively. If t is the next possible time where at least one job can be scheduled (see lines 3 and 11), the job with the greatest delivery time is chosen among those with a ready date compatible with t (see lines 5 till 7).

Algorithm 1: H_0

```
Input  : $n$ jobs with release dates $r_j$, processing times $p_j$ and delivery times $q_j$ $j = 1, \ldots, n$
Output: A schedule of the jobs on a single machine
1  $\overline{U} = \{1, \ldots, n\}$ the set of jobs not yet scheduled
2  Schedule $\leftarrow \emptyset$
3  $t = \min_{j \in \overline{U}} r_j$
4  while $\overline{U} \neq \emptyset$ do
5      $J_1 = \{j \in \overline{U} \mid r_j \leq t\}$
6      $J_2 = \{j \in J_1 \mid q_j = \max_{i \in J_1} q_i\}$
7      $J_3 = \{j \in J_2 \mid r_j = \min_{i \in J_2} r_i\}$
8      Schedule one job $i \in J_3$ at time $t$
9      Schedule $\leftarrow$ Schedule $\bigcup \{i\}$
10     $\overline{U} = \overline{U} \setminus \{i\}$
11     $t = \max(t + p_i, \min_{j \in \overline{U}} r_j)$
12 return Schedule
```

3.2.2 *The heuristic H_{1a}*

The pseudo-code of heuristic H_{1a} is shown in Algorithm 2. This heuristic determines progressively the order of the jobs in the schedule, according to the following principle. The minimal value of the ready dates and delivery times of the remaining jobs is determined (see line 6). If this value is a ready date (delivery time) the corresponding job is placed at the first (last) position available in the order (see lines 7,10,11 and 12 (lines 8,14,15 and 16)).

3.2.3 *The heuristic H_{1b}*

The pseudo-code of heuristic H_{1b} is shown in Algorithm 3. In the same way, this heuristic determines progressively the order of the jobs in the schedule, according to the following principle. The maximal value of the ready dates and delivery times of the remaining jobs is determined (see line 6). If this value is a ready date (delivery time) the corresponding job is placed at the last (first) position available in the order (see lines 8,14,15 and 16 (lines 7,10,11 and 12)).

3.2.4 *The heuristic H_2*

The pseudo-code of heuristic H_2 is shown in Algorithm 4. 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 one strict inequality. Clearly if $i \succ j$, i must be scheduled before j.

First, the jobs are ordered in non-decreasing ready dates (see line 1) and the heuristic builds the schedule progressively. At each iteration, the set ND of non-dominated jobs is updated (see lines 7 and 8). If t is the new possible time where at least one job can be scheduled (see lines 4 and 25), some jobs can be eliminated of ND because they are less interesting to be scheduled (see lines 9 till 12). Among the remaining jobs of ND, a pairwise comparison of two jobs is made based on the slacks between their respective ready dates and delivery times (see lines 13,14 and 15) and at each comparison one job is eliminated of ND (see lines 16 till 21). The last job remaining in ND is chosen to be scheduled (see lines 22 and 23).

3.3 *Illustrations and comparison of the heuristics*

These four heuristics will first be illustrated on three didactic instances. For each instance, at least one heuristic will not provide the optimal schedule. So the aim of these three instances will be first to compare the mechanisms of the four heuristics and

Algorithm 2: H_{1a}

```
Input  : $n$ jobs with release dates $r_j$, processing times $p_j$ and delivery times $q_j$ $j = 1, \dots, n$
Output: An order of the jobs on a single machine
$\overline{U} = \{1, \dots, n\}$ the set of jobs not yet ordered
Order $\leftarrow \emptyset$
$p = 1$
$q = n$
while $\overline{U} \neq \emptyset$ do
    $\delta = \min\{r_j, q_j \; j \in \overline{U}\}$
    $P = \{j \in \overline{U} \mid r_j = \delta\}$
    $Q = \{j \in \overline{U} \mid q_j = \delta\}$
    if $P \neq \emptyset$ then
        $I = \{j \in P \mid q_j = \max_{i \in P} q_i\}$
        Place one job $i \in I$ at position $p$
        $p \leftarrow p + 1$
    else
        $I = \{j \in Q \mid r_j = \max_{i \in Q} r_i\}$
        Place one job $i \in I$ at position $q$
        $q \leftarrow q - 1$
    Order $\leftarrow$ Order $\bigcup \{i\}$
    $\overline{U} = \overline{U} \setminus \{i\}$
return Order
```

Algorithm 3: H_{1b}

```
Input  : $n$ jobs with release dates $r_j$, processing times $p_j$ and delivery times $q_j$ $j = 1, \dots, n$
Output: An order of the jobs on a single machine
$\overline{U} = \{1, \dots, n\}$ the set of jobs not yet ordered
Order $\leftarrow \emptyset$
$p = 1$
$q = n$
while $\overline{U} \neq \emptyset$ do
    $\delta = \max\{r_j, q_j \; j \in \overline{U}\}$
    $P = \{j \in \overline{U} \mid q_j = \delta\}$
    $Q = \{j \in \overline{U} \mid r_j = \delta\}$
    if $P \neq \emptyset$ then
        $I = \{j \in P \mid r_j = \min_{i \in P} r_i\}$
        Place one job $i \in I$ at position $p$
        $p \leftarrow p + 1$
    else
        $I = \{j \in Q \mid q_j = \min_{i \in Q} q_i\}$
        Place one job $i \in I$ at position $q$
        $q \leftarrow q - 1$
    Order $\leftarrow$ Order $\bigcup \{i\}$
    $\overline{U} = \overline{U} \setminus \{i\}$
return Order
```

Algorithm 4: H_2

```
Input  : $n$ jobs with release dates $r_j$, processing times $p_j$ and delivery times $q_j$ $j = 1, \dots, n$
Output: A schedule of the jobs on a single machine
1  Order the $n$ jobs such that $r_1 \leq r_2 \leq \dots \leq r_n$ with $q_j \geq q_{j+1}$ when $r_j = r_{j+1}$
2  $\overline{U} = \{1, \dots, n\}$ the set of jobs not yet scheduled
3  Schedule $\leftarrow \emptyset$
4  $t = \min_{j \in \overline{U}} r_j$
5  $k = 1$
6  while $\overline{U} \neq \emptyset$ do
7      Determine $D_i = \{j \in \overline{U} \mid j < i,\ q_j \geq q_i\}\ \forall i \in \overline{U}$
8      $ND = \{i \in \overline{U} \mid D_i = \emptyset\}$
9      $l = \arg\max_j \{j \in ND \mid r_j \leq t\}$
10     $ND \leftarrow ND \setminus \{j \in ND \mid j < l\}$
11     Determine $E_i = \{j \in ND \mid j < i,\ r_i + p_i \leq r_j\}\ \forall i \in ND$
12     $ND \leftarrow ND \setminus (\bigcup_{i \in ND} E_i)$
13     foreach $(i, j) \in ND$ with $j > i$ do
14         Calculate the lost $r_j - r_i$
15         Calculate the gain $q_j - q_i$
16     while $|ND| > 1$ do
17         Determine $(m^*, n^*) \in ND$ such that $q_{n^*} - q_{m^*} = \min_{(i,j) \in ND,\ j>i} (q_j - q_i)$
18         if $r_{n^*} - r_{m^*} < q_{n^*} - q_{m^*}$ then
19             $ND \leftarrow ND \setminus \{m^*\}$
20         else
21             $ND \leftarrow ND \setminus \{n^*\}$
22     Schedule the remaining job $i \in ND$ at the place $k$
23     Schedule $\leftarrow$ Schedule $\bigcup \{i\}$
24     $\overline{U} = \overline{U} \setminus \{i\}$
25     $t = \max(t + p_i, \min_{j \in \overline{U}} r_j)$
26     $k \leftarrow k + 1$
27 return Schedule
```

to analyse their respective characteristics (see Section 3.3.4) and secondly to illustrate the performance of the improvement algorithm proposed in Section 3.4.

Let us recall here the notion of critical path inside a schedule (Carlier 1982). Such critical path $CP = \{a, \dots, p\}$ begins with a first job a and finishes with a last job p such that the final date $M = C_p + q_p$. Inside CP, there does not exist any empty time. The path CP begins at time T_a and can be preceded by an empty time E_a: if $E_a \neq 0$ then $T_a = r_a$, if $E_a = 0$ then $T_a = r_a = 0$.

It appears from the study of Carlier that the schedule is optimal if simultaneously the relations

$$r_a = \min_{i \in \{a,\dots,p\}} r_i \tag{1}$$

and

$$q_p = \min_{i \in \{a,\dots,p\}} q_i \tag{2}$$

are satisfied. Relations (1) and (2) are thus sufficient (but not necessary) conditions of optimality.

The mechanisms of these heuristics will be compared and their respective characteristics will be analysed.

3.3.1 *Instance 1*

AQ10 This first instance comes from Carlier (1982) (Tables 1–5).

Table 1. Data of instance 1.

j	1	2	3	4	5	6	7
r_j	10	13	11	20	30	0	30
p_j	5	6	7	4	3	6	2
q_j	7	26	24	21	8	17	0

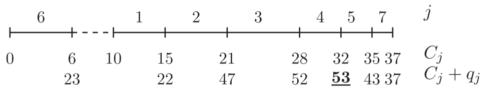
6 1 2 3 4 5 7 j
0 6 10 15 21 28 32 35 37 C_j
23 22 47 52 **53** 43 37 $C_j + q_j$

Figure 1. Instance 1: schedule with H_0.

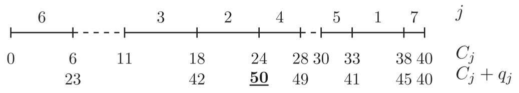
6 3 2 4 5 1 7 j
0 6 11 18 24 28 30 33 38 40 C_j
23 42 **50** 49 41 45 40 $C_j + q_j$

Figure 2. Instance 1: schedule with H_{1a}.

This schedule is optimal as relations (1) and (2) are satisfied in $CP = \{3, 2\}$.

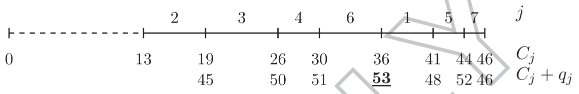
2 3 4 6 1 5 7 j
0 13 19 26 30 36 41 44 46 C_j
45 50 51 **53** 48 52 46 $C_j + q_j$

Figure 3. Instance 1: schedule with H_{1b}.

Table 2. Dominance relation for instance 1.

	1	2	3	4	5	6	7
1	–						$\succ$
2		–		$\succ$	$\succ$		$\succ$
3			–	$\succ$	$\succ$		$\succ$
4				–	$\succ$		$\succ$
5					–		$\succ$
6	$\succ$				$\succ$	–	$\succ$
7							–

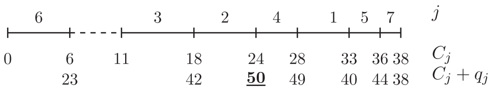
6 3 2 4 1 5 7 j
0 6 11 18 24 28 33 36 38 C_j
23 42 **50** 49 40 44 38 $C_j + q_j$

Figure 4. Instance 1: schedule with H_2.

This schedule is optimal as relations (1) and (2) are satisfied in $CP = \{3, 2\}$.
We remark that the four heuristics generate four different schedules.

3.3.2 *Instance 2*

Table 3. Data of instance 2.

j	1	2	3	4	5	6	7	8	9
r_j	2	8	11	1	5	5	0	6	7
p_j	3	2	3	3	2	9	6	2	6
q_j	7	3	0	5	3	0	1	2	1

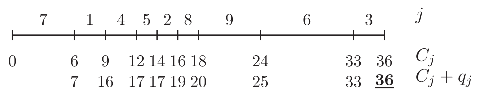
7 1 4 5 2 8 9 6 3 j
0 6 9 12 14 16 18 24 33 36 C_j
7 16 17 17 19 20 25 33 **36** $C_j + q_j$

Figure 5. Instance 2: schedule with H_0.

This schedule is optimal as relations (1) and (2) are satisfied in $CP = \{7, 1, 4, 5, 2, 8, 9, 6, 3\}$.

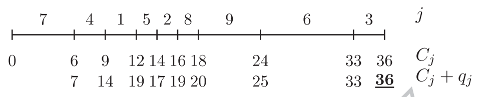
7 4 1 5 2 8 9 6 3 j
0 6 9 12 14 16 18 24 33 36 C_j
7 14 19 17 19 20 25 33 **36** $C_j + q_j$

Figure 6. Instance 2: schedule with H_{1a}.

This schedule is optimal.

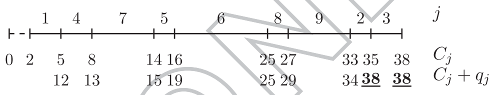
1 4 7 5 6 8 9 2 3 j
0 2 5 8 14 16 25 27 33 35 38 C_j
12 13 15 19 25 29 34 **38** **38** $C_j + q_j$

Figure 7. Instance 2: schedule with H_{1b}.

Table 4. Dominance relation for instance 2.

	1	2	3	4	5	6	7	8	9
1	–	≻	≻		≻	≻		≻	≻
2		–	≻						
3			–						
4		≻	≻	–	≻	≻		≻	≻
5		≻	≻		–	≻		≻	≻
6			≻			–			
7			≻			≻	–		≻
8			≻					–	≻
9			≻						–

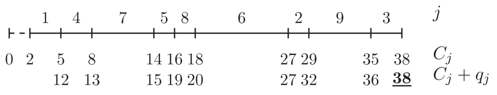

Figure 8. Instance 2: schedule with H_2.

We remark again that the four heuristics generate four different schedules.

3.3.3 *Instance 3*

Table 5. Data of instance 3.

j	1	2	3	4	5
r_j	2	8	11	5	6
p_j	3	2	3	2	2
q_j	7	3	0	3	2

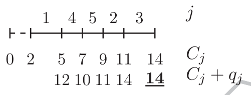

Figure 9. Instance 3: schedule with H_0.

This schedule is optimal as relations (1) and (2) are satisfied in $CP = \{1, 4, 5, 2, 3\}$. Both heuristics H_{1b} and H_2 provide the same optimal schedule of Figure 9.

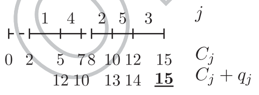

Figure 10. Instance 3: schedule with H_{1a}.

From Figures 1 till 10, and relations (1) and (2), it appears that

- for instance 1, H_1a and H_2 provide an optimal schedule, with $M = 50$;
- for instance 2, H_0 and H_1a provide an optimal schedule, with $M = 36$;
- for instance 3, H_0, H_1b and H_2 provide an optimal schedule, with $M = 14$.

3.3.4 *Comparison of the heuristics*

- A first distinction can be made between $\{H_0, H_2\}$ on one side and $\{H_{1a}, H_{1b}\}$ on another side. H_0 and H_2 provide immediately a schedule taking into account the next possible time t when at least one job can be scheduled. It is not the case for H_{1a} and H_{1b} which define first an order of jobs, independently of this time t, in which the jobs must be scheduled in a second step. Effectively in these two heuristics, the jobs of the set $A = \{j \mid r_j \leq q_j\}$ are placed first followed by the jobs of the set $B = \{j \mid r_j > q_j\}$. In H_{1a}, the jobs of A are ranked in increasing order of r_j and the jobs of B are ranked in decreasing order of q_j. In H_{1b}, the jobs of A are ranked in decreasing order of q_j and the jobs of B are ranked in increasing order of r_j. A consequence of this distinction will be described in Section 3.4.
- It is easy to verify that H_0, H_{1a} and H_{1b} also satisfy the dominance relation used in H_2. Thus, in all the heuristics, if $r_i \leq r_j$ and $q_i \geq q_j$, with at least a strict inequality, the job i will preceed the job j.

- Consequently, the main distinction between the four heuristics concerns the choice between two non-dominated jobs, i.e. between two jobs i and j with $r_i \leq r_j$ and $q_i \leq q_j$. Contrary to H_2, in the three other heuristics the slacks $r_j - r_i$ and $q_j - q_i$ are not used to determine which job to schedule first.

  In H_0 $\left[\begin{array}{l}\text{- if } r_i \leq r_j \leq t, \text{ the job } j \text{ is placed first}\\ \text{- if } r_i \leq t < r_j, \text{ the job } i \text{ is placed first}\end{array}\right.$

  In H_{1a} $\left[\begin{array}{l}\text{- if } \min(r_i, q_i) = r_i \text{ the job } i \text{ is placed first}\\ \text{- if } \min(r_i, q_i) = q_i \text{ the job } j \text{ is placed first}\end{array}\right.$

  In H_{1b} $\left[\begin{array}{l}\text{- if } \max(r_i, q_i) = q_i \text{ the job } i \text{ is placed first}\\ \text{- if } \max(r_i, q_i) = r_i \text{ the job } j \text{ is placed first}\end{array}\right.$

- Finally, we can also note that
  - an advantage of H_0 is that relation (1) is always satisfied in the critical path, which is not the case with the three other heuristics
  - a drawback of H_{1b} (confirmed by the numerical experiments of Section 3.5) is to not take enough into account the ready dates in the order of the jobs of the set A yet placed at the beginning of the schedule.

3.4 *An improvement algorithm*

We now propose an algorithm H^+ to improve the final date M provided with any heuristic $H \in \{H_0, H_{1a}, H_{1b}, H_2\}$. This improvement algorithm is based on the critical path inside the current schedule. We use the notations introduced in Section 3.3: $CP = \{a, \ldots, p\}$, $M = C_p + q_p$, $E_a \neq 0 \rightarrow T_a = r_a$, $E_a = 0 \rightarrow T_a = r_a = 0$.

This algorithm will be illustrated on the three didactic instances in all the cases when the schedule obtained by a heuristic H is not optimal.

3.4.1 *Algorithm H^+*

Related to relations (1) and (2), the main idea of H^+ consists of analysing if the final data M can be decreased either placing a job $j \in CP$ with $r_j < r_a$ before CP or placing a job $j \in CP$ with $q_j < q_p$ after CP.

The pseudo-code of the improvement algorithm H^+ is shown in Algorithm 5. It starts with an initial current schedule (see line 1). For each job $j \in CP$, we first test if $r_j < r_a$ (lines 7 and 8). For all such jobs j, we determine the new schedule placing j just before a in the current schedule (line 9) and in each time where the final date is improved, the new schedule becomes the best schedule (lines 10 and 11). When there is no more possibility to improve M in this manner (line 13), for each job $j \in CP$ we then test if $q_j < q_p$ (lines 14 and 15). For all such jobs j, we determine the new schedule placing j just after p in the current schedule (line 16) and in case where the final date is improved, the new schedule becomes the best schedule (lines 17 and 18).

3.4.2 *Illustrations of H^+*

We illustrate the algorithm H^+ in the five cases where a heuristic H does not provide an optimal schedule for the three didactic instances of Section 3.3.

Instance 1: schedule H_0^+

H_0 provides the non-optimal schedule of Figure 1.

$a = 1$, $p = 4$, $CP = \{1, 2, 3, 4\}$, $\{j \in CP \mid r_j < r_a\} = \emptyset$ and $\{j \in CP \mid q_j < q_p\} = \{1\}$.

Placing the job 1 after the job 4, we obtain the schedule of Figure 11.

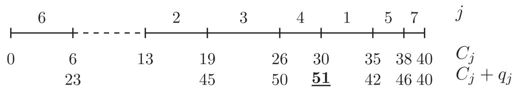

Figure 11. Instance 1: first iteration of $H0+$.

$a = 2$, $p = 4$, $CP = \{2, 3, 4\}$, $\{j \in CP \mid r_j < r_a\} = \{3\}$ and $\{j \in CP \mid q_j < q_p\} = \emptyset$.

Placing the job 3 before the job 2, we obtain the optimal schedule of Figure 4.

Algorithm 5: H^+

```
Input  : Schedule obtained from a heuristic
Output: Best schedule found
BestSchedule ← Schedule
Improve ← True
while Improve = True do
    Improve ← False
    CurrentSchedule ← BestSchedule
    Find the critical path $CP = \{a, \dots, p\}$ of CurrentSchedule
    foreach $j \in CP$ do
        if $r_j < r_a$ then
            NewSchedule ← place job $j$ before job $a$ in CurrentSchedule
            if final date of NewSchedule < final date of BestSchedule then
                BestSchedule ← NewSchedule
                Improve ← True
    if Improve = False then
        foreach $j \in CP$ do
            if $q_j < q_p$ then
                NewSchedule ← place job $j$ after job $p$ in CurrentSchedule
                if final date of NewSchedule < final date of BestSchedule then
                    BestSchedule ← NewSchedule
                    Improve ← True
Schedule ← BestSchedule
return Schedule
```

Instance 1: schedule H_{1b}^+
H_{1b} provides the non-optimal schedule of Figure 3.
$a = 2$, $p = 6$, $CP = \{2, 3, 4, 6\}$, $\{j \in CP \mid r_j < r_a\} = \{3, 6\}$ and $\{j \in CP \mid q_j < q_p\} = \emptyset$.
The best new schedule is obtained placing the job 6 before the job 2. It corresponds to the schedule of Figure 11. By a second iteration, identical as the one of H_0^+ above, we obtain the optimal schedule of Figure 4.
Instance 2: schedule H_{1b}^+
H_{1b} provides the non-optimal schedule of Figure 7.
$a = 1$, $p = 3$, $CP = \{1, 4, 7, 5, 6, 8, 9, 2, 3\}$, $\{j \in CP \mid r_j < r_a\} = \{4, 7\}$ and $\{j \in CP \mid q_j < q_p\} = \emptyset$.
The best new schedule is obtained placing the job 7 before the job 1. We obtain the optimal schedule of Figure 12.

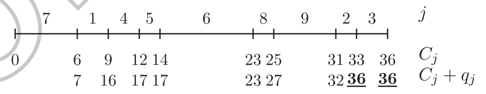

Figure 12. Instance 2: optimal schedule with H_{1b}^+.

Instance 2: schedule H_2^+
H_2 provides the non-optimal schedule of Figure 8.
$a = 1$, $p = 3$, $CP = \{1, 4, 7, 5, 8, 6, 2, 9, 3\}$, $\{j \in CP \mid r_j < r_a\} = \{4, 7\}$ and $\{j \in CP \mid q_j < q_p\} = \emptyset$.
As above, the best new schedule is obtained placing the job 7 before the job 1. We obtain the optimal schedule of Figure 13.

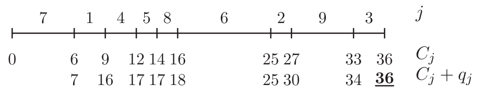
7 1 4 5 8 6 2 9 3 j
0 6 9 12 14 16 25 27 33 36 C_j
7 16 17 17 18 25 30 34 **36** $C_j + q_j$

Figure 13. Instance 2: optimal schedule with H_2^+.

Instance 3: schedule H_{1a}^+
H_{1a} provides the non-optimal schedule of Figure 10.
$a = 2$, $p = 3$, $CP = \{2, 5, 3\}$, $\{j \in CP \mid r_j < r_a\} = \{5\}$ and $\{j \in CP \mid q_j < q_p\} = \emptyset$.
Placing the job 5 before the job 2, we obtain the optimal schedule of Figure 9.
So, on these three didactic instances, all algorithms H^+ provide always an optimal schedule.

3.5 *Numerical experiments*

A large set of 40 instances (four different instances for each $n \in \{10, 50\}$ and eight different instances for each $n \in \{100, 200, 500, 1000\}$) is randomly generated by the way proposed by Carlier (1982). The processing times are generated with a uniform distribution in interval [1, 50]; the release dates and the delivery times are generated with a uniform distribution in interval $[1, n\,K]$, with $K \in \{1, 20, 40, 120\}$ (each value of K is used one time for the four instances with $n \in \{10, 50\}$ and two times for the eight instances with $n \in \{100, 200, 500, 1000\}$).

Table 6 indicates the number of times the best value is obtained among the four algorithms H_0, H_{1a}, H_{1b}, H_2. Table 7 indicates the number of times the best value is obtained among the four algorithms H_0^+, H_{1a}^+, H_{1b}^+, H_2^+. In each of these tables, the first line corresponds to eight 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 6. Results of the 4 heuristics H.

n	H_0	H_{1a}	H_{1b}	H_2
10, 50	5	7	0	7
100,200	9	16	0	13
500,1000	13	16	0	12
Total	27	39	0	32

Table 7. Results of the four heuristics $H+$.

n	H_0^+	H_{1a}^+	H_{1b}^+	H_2^+
10, 50	8	8	8	8
100,200	16	16	16	16
500,1000	15	16	16	16
Total	39	40	40	40

From Table 6, it appears

- that the performance of $H1b$ is really bad. As indicated at the end of Section 3.3.4, it appears that this heuristic does not take enough into account the ready dates of the jobs, in particular for the jobs of the set A placed at the beginning of the schedule provided by this heuristic.
- for the 40 instances, $H1a$ has really the best performance followed by $H2$ and $H0$.

From Table 7, it appears

- that the algorithm H^+ is really powerful independently of the heuristic H used; effectively among the 160 experiments (40 instances with 4 heuristics) the best value is obtained 159 times. It is remarkable that despite the very poor performance of H_{1b}, H_{1b}^+ always obtains the best value.

- that the unique exception of no best value obtained with H_0^+ is due to the distinction described at the beginning of Section 3.3.4. We recall that, contrary to H_{1a} and H_{1b}, providing an order of jobs, H_0 and H_2 generates immediately a schedule using the time t when at least a job can be scheduled. A consequence is that when a job of the critical path is moved either before the job a or either after the job p, new idle time can appear. In such case, the time t can be modified and the schedule of the jobs can no more respect the rule of these two heuristics, because H^+ does not change the order of the jobs except the move of the job j. Analysing in details the instance of this exception (H_0^+ with $n = 500$) it appears that due to the new idle time appearing, the order of the jobs located after the critical path must be modified to respect the rule of H_0. Such situation cannot appear with H_{1a}^+ and H_{1b}^+.

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.

In the description of the method, we will use the following notations:

- m_k $k = 1, \dots, m$ the machine of level k.
- J_k $k = 1, \dots, m$ the subset of jobs which can be assigned to machines m_l with $l = 1, \dots, k$, with $J_1 = \{1, \dots, n\}$; $J_{k+1} \subset J_k$.
- P_k $k = 1, \dots, m$ the penalty incurred each time a job is assigned to machine m_k ($P_1 = 0$).
- I_k $k = 1, \dots, m$ the subset of jobs assigned to machine m_k in a solution, with $I_k \subseteq J_k$.
- M_k $k = 1, \dots, m$ the final date of the jobs I_k on machine m_k defined by

$$M_k = \max_{j \in I_k} (C_j + q_j)$$

- M the final date of a solution $M = \max_{k=1,\dots,m} M_k$.
- m^*, called dominating machine, which is the machine with the largest index k such that

$$M_k = M$$

- The two objectives to minimise are the final date M and the total penalty $P = \sum_{k=2}^{m} P_k\,|I_k|$

Remark To assign a subset I_k of jobs to a machine k, we always use the algorithm H_{1a}^+ of the Section 3, denoted by 'Heuristic(I_k)'. It generates the final date M_k and the critical path C_k of the corresponding schedule.

To implement the method, we use a stack data structure (Last-In-First-Out (LIFO) list). Such data structure appears generally in all Depth First Search algorithms. We use the following primitive operations:

Primitive	Operation
Stack	Create an empty stack
isEmpty	Determine whether the stack is empty
push	Add an item to the top of the stack
pop	Remove the item most recently added
peek	Retrieve the item most recently added

Each item contains three informations: an action, the index of a job transferred and the index of the machine from which the job is removed. Three different actions are considered: NoBT (doing nothing), BT1 (starting a backtracking of type 1) and BT2 (starting a backtracking of type 2).

4.1 *First phase: depth first phase*

The pseudo-code of this phase is described in Algorithm 6.

The initial solution corresponds to the case $I_1 = \{1, \dots, n\}$, $I_k = \emptyset, k = 2, \dots, m$. The Heuristic schedules all the jobs on machine m_1 so that $P = 0,\ M = M_1,\ M_k = 0, k = 2, \dots, m,\ m^* = 1$. PE contains the solutions (P, M) generated by the method (see lines 1 till 7). P is called Pen in the pseudo-code. The first item is added in the stack with no job transferred ($i^* = 0$) and ($m^* = 1$) (line 8).

Then, each iteration consists to generate a new solution by the transfer of a job i of J_{m^*+1} included in the critical path C_{m^*} from the dominating machine m^* to the machine $m^* + 1$ (see line 13). The first phase will finish either when $m^* = m$ (see line 10) or if there is no more job able to be transferred (see lines 14 and 15). For any possible job to transfer, the Heuristic determines the new final date and critical paths of machines m^* and $m^* + 1$ (lines 17 till 19) and the best job i^* to transfer which generates the minimum of the maximal value of these two final dates (line 20).

Remark on line 20 (best job i^*)

In case of several jobs i corresponding to the same value

$$N = \min_i \left(\max \left(M^{(i)}_{m^*}, M^{(i)}_{m^*+1}\right)\right)$$

preference is given

- first to jobs with $N = M^{(i)}_{m^*+1}$ if $m^* + 1 < m$.
- secondly to jobs with the minimal value on the other machine.

The values corresponding to this transfer are updated (lines 21 till 23) and the new solution is placed inside PE (line 24). If the level of the current dominating machine is greater than the level of the machine corresponding to the last item added in the stack, this last item is updated for the action Backtracking 1 (BT1) at the phase 2 (lines 25 till 27). The new item is added in the stack with the job i^* and the machine m^* (line 28). The new dominating machine is updated (line 29).

At the end of the first phase, the solutions inside PE are filtered to remove the dominated solutions (see line 30).

4.2 *Second phase: the backtracking phase*

Algorithm 7 manages the general organisation of this phase and the call of the two types of backtrackings, noted BT1 (described in Algorithm 8) and BT2 (described in Algorithm 9).

All the items registered in the stack with action BT1 at phase 1 correspond to solutions where the transfer of a job i^* from machine m^* till $m^* + 1$ increases the index of the new dominating machine.

In any case, the job i^* is first replaced on machine m^* coming back from the machine $m^* + 1$, and the penalty is updated (lines 8 and 9). The backtracking BT1 is then applied (line 14). When the last item added in the stack does not already correspond to action BT1 and its machine is inferior to $m - 1$, the backtracking BT2 will be applied to this item (lines 11 and 13).

4.2.1 *The backtracking BT1 (Algorithm 8)*

Two successive iterations are made, for the next reason. If it is true that the transfer of i^* from m^* till $m^* + 1$ in the phase 1 generates the best possible solution at this iteration, nevertheless it is possible that two successive iterations (with at the first one the transfer of a job different than i^* from m^* till $m^* + 1$) will produce a better solution that the one obtained at phase 1.

So at the first iteration, all the possible jobs $i \in I \setminus \{i^*\}$ to transfer from m^* till $m^* + 1$ are considered and the new final dates $M^{(i)}_{m^*}$ and $M^{(i)}_{m^*+1}$ are obtained with the Heuristic (lines 4 till 6).

Then a second iteration will transfer a job $j(i)$ either from $m^*(= mm^*)$ till $m^* + 1$ if $M^{(i)}_{m^*} > M^{(i)}_{m^*+1}$ or from $m^* + 1(= mm^*)$ till $m^* + 2$ otherwise (lines 7 till 15). The best $j^*(i)$ is first selected for each $i \in I$ and the best pair $(i^*, j^*(i^*))$ (with a new i^*) is then chosen (according to the rule described in the remark in Section 4.1)(line 16). The values corresponding to this double transfer are updated (lines 17 till 22). If the new generated solution is dominated by a solution of PE, the backtracking BT1 is finished (lines 23 and 24). Otherwise the solution is registered in PE and the phase 1 will be applied to the corresponding solution (lines 25 and 26).

4.2.2 *The backtracking BT2 (Algorithm 9)*

The idea of the backtracking 2 is to replace an iteration of phase 1, where $m^* < m - 1$, by the transfer of a job directly from m^* to $m^* + 2$.

Thus in Algorithm 9, a new solution is generated by the transfer of a job i from the machine m^* to $m^* + 2$ (lines 5 and 6) and the best job i^* is chosen (according to the rules described in the remark of Section 4.1) (line 7). The values corresponding to the transfer are updated (lines 8 till 12). If the new generated solution is dominated by a solution of PE, the backtracking BT2 is finished (lines 13 and 14). Otherwise the solution is registered in PE and the phase 1 will be applied to the corresponding solution (lines 15 and 16).

Algorithm 6: Phase 1

```
Input  : $n$ jobs with release dates $r_j$, processing times $p_j$ and delivery times $q_j$ $j = 1, \dots, n$ and $m$ machines with $J_m \subset \dots \subset J_1$
Output: A list of potential efficient solutions

1  PE $\leftarrow \emptyset$
2  STACK S
3  $I_1 = \{1, \dots, n\},\ I_2 = \emptyset, \dots, I_m = \emptyset$
4  $(M_1, CP_1) \leftarrow Heuristic(I_1)$
5  $Pen = 0$
6  $PE \leftarrow PE \bigcup \{(Pen, M_1)\}$
7  $m^* = 1$
8  $PUSH(S, [NoBT, 0, m^*])$
9  while IsEMPTY(S) = False do
10     if $m^* = m$ then
11         Call Phase 2
12     else
13         $I = \{i \in CP_{m^*} \bigcap J_{m^*+1}\}$
14         if $I = \emptyset$ then
15             Call Phase 2
16         else
17             foreach $i \in I$ do
18                 $(M_{m^*}^{(i)}, CP_{m^*}^{(i)}) \leftarrow Heuristic(I_{m^*} \setminus \{i\})$
19                 $(M_{m^*+1}^{(i)}, CP_{m^*+1}^{(i)}) \leftarrow Heuristic(I_{m^*+1} \bigcup \{i\})$
20             Select the best job $i^* \in I$ with $i^* = \arg\min_{i \in I} \{\max(M_{m^*}^{(i)}, M_{m^*+1}^{(i)})\}$
21             Transfer job $i^*$ on machine $m^* + 1 : I_{m^*} \leftarrow I_{m^*} \setminus \{i^*\}, I_{m^*+1} \leftarrow I_{m^*+1} \bigcup \{i^*\}$
22             $Pen \leftarrow Pen + P_{m^*+1} - P_{m^*}$
23             $M = \max\{M_1, \dots, M_m\}$
24             $PE \leftarrow PE \bigcup \{(Pen, M)\}$
25             $PEEK(S, [action, job, machine])$
26             if $m^* > machine$ then
27                 $action \leftarrow BT1$
28             $PUSH(S, [NoBT, i^*, m^*])$
29             $m^* = \arg\max_i \{M_i = M\ i = 1, \dots, m\}$
30 Remove dominated solutions in $PE$
31 return PE
```

4.3 *An illustration*

It concerns an instance with 20 jobs and 3 machines. The data of this instance are given in Table 8 with $J_3 = \{17, \dots, 20\} \subset J_2 = \{7, \dots, 20\} \subset J_1 = \{1, \dots, 20\}$. The penalties are fixed to $P_1 = 0$, $P_2 = 1$ and $P_3 = 2$.

4.3.1 *Phase 1*

In Table 9, we describe each iteration of the first phase of the heuristic indicating in the first column the solution (P, M) obtained, in the second column (M_1, M_2, M_3) with M^* in bold type, in the next three columns the information contained in the stack and in the last column the orders of the jobs on the three machines which are presented in the format ..//..//.., with i^* in bold type.

The first phase finishes because there exists no job to transfer from m_2 to m_3. There are four situations where a backtracking can be made, respectively, for solutions (6, 67), (8, 65) and (10, 58) for a backtracking of type 1 and for solution (5, 76) for a backtracking of type 2.

Algorithm 7: Phase 2

```
Input  : The stack
Output: An updated stack where items are removed until an action is required
1  Exit ← $False$
2  while $Exit = False$ do
3      if $IsEMPTY(S) = True$ then
4          Exit ← $True$
5      else
6          $POP(S, [action, i^*, m^*])$
7          if $i^* \neq 0$ then
8              Restore job $i^*$ on machine $m^*$ : $I_{m^*} \leftarrow I_{m^*} \bigcup \{i^*\}$, $I_{m^*+1} \leftarrow I_{m^*+1} \setminus \{i^*\}$
9              $Pen \leftarrow Pen + P_{m^*} - P_{m^*+1}$
10         if $action = BT1$ then
11             $PEEK(S, [action, job, machine])$
12             if $machine < m - 1$ and $action \neq BT1$ then
13                 $action \leftarrow BT2$
14             Call BackTracking 1
15             Exit ← $ContinuePhase1$
16         else
17             if $action = BT2$ then
18                 Call BackTracking 2
19                 Exit ← $ContinuePhase1$
20 return S
```

4.3.2 *Phase 2*

We only present in Table 10 some examples of backtrackings generating interesting solutions. Some solutions can be obtained by different backtrackings (see solution (12,51)).

As the dimension of this instance is small, it is possible, by a long but complete enumeration, to obtain the exact Pareto front. This Pareto front contains the following 15 solutions: (0,124), (1,114), (2,104), (3,94), (4,85), (5,76), (6,67), (7,64), (8,60), (9,59), (10,54), (11,53), (12,51), (13,50), (14,49). For this instance, 14 solutions have been found by the heuristic and only one solution (7,66) is dominated. Effectively, the following solution (7,64) appears in the Pareto front with $M_1 = 64$, $M_2 = 64$ and $M_3 = 0$ and the following orders of the jobs on the three machines 1-17-16-5-8-3-20-4-7-13-9-2 // 10-19-14-15-18-11-12 //.

Comparing this solution with the solution (7,66) obtained by the first phase of the heuristic (see Table 9), it appears that the job 16, transferred from m_1 to m_2 at the first iteration (see solution (1,114) in Table 9) goes back to m_1 and is replaced on m_2 by the jobs 11 and 12. Clearly such possibility is not taken into account in the heuristic due to the very high combinatorial aspect of such mechanism. In the backtracking of type 1 of the heuristic, only the last job transferred to m_2 at the last iteration is considered to be replaced on m_2. Nevertheless, it gives a possible direction of research to improve the performance of the heuristic in the future.

To conclude with this first example, comparing to the 15 solutions of the exact Pareto front

- the first phase generates 12 solutions; 58% are equal to those of the exact Pareto front and the others are dominated with a maximal error on M equal to 8%.
- the heuristic with its two phases generates 15 solutions; 93% are equal to those of the exact Pareto front and the other is dominated with an error on M equal to 3%.

5. Numerical experiments

5.1 *Instances with 20 jobs and 3 machines*

We randomly generate two sets (noted set A and set B) of 10 instances with 20 jobs. In the first, the processing times are generated with an uniform distribution in the interval [1,10], the release dates and delivery times with an uniform distribution in the interval [1,20]. In the second, based on Carlier (1982), the processing times are generated with a uniform

Algorithm 8: BackTracking 1

Input : A partition of the jobs on the machines
Output: A new partition of the jobs on the machines by doing two consecutive iterations of Phase 1

```
1  $ContinuePhase1 \leftarrow True$
2  $I = \{i \in CP_{m^*} \bigcap J_{m^*+1}\} \setminus \{i^*\}$
3  if $I \neq \emptyset$ then
4      foreach $i \in I$ do
5          $(M^{(i)}_{m^*}, CP^{(i)}_{m^*}) \leftarrow Heuristic(I_{m^*} \setminus \{i\})$
6          $(M^{(i)}_{m^*+1}, CP^{(i)}_{m^*+1}) \leftarrow Heuristic(I_{m^*+1} \bigcup \{i\})$
7          if $M^{(i)}_{m^*} > M^{(i)}_{m^*+1}$ then
8              $mm^* = m^*$
9          else
10             $mm^* = m^* + 1$
11         $J = \{j(i) \in CP^{(i)}_{mm^*} \bigcap J_{mm^*+1}\}$
12         if $J \neq \emptyset$ then
13             foreach $j(i) \in J$ do
14                 $(M^{(i)(j(i))}_{mm^*}, CP^{(i)(j(i))}_{mm^*}) \leftarrow Heuristic(I_{mm^*} \setminus \{j(i)\})$
15                 $(M^{(i)(j(i))}_{mm^*+1}, CP^{(i)(j(i))}_{mm^*+1}) \leftarrow Heuristic(I_{mm^*+1} \bigcup \{j(i)\})$
16             Select the best job $j^*(i) \in J$ and the best job $i^* \in I$
17             Transfer job $i^*$ on machine $m^* + 1 : I_{m^*} \leftarrow I_{m^*} \setminus \{i^*\}$,$I_{m^*+1} \leftarrow I_{m^*+1} \bigcup \{i^*\}$
18             Transfer job $j^*(i^*)$ on machine $mm^* + 1$ :
19             $I_{mm^*} \leftarrow I_{mm^*} \setminus \{j^*(i^*)\}$,$I_{mm^*+1} \leftarrow I_{mm^*+1} \bigcup \{j^*(i^*)\}$
20             $PUSH(S, [NoBT, i^*, m^*])$
21             $PUSH(S, [NoBT, j^*(i^*), mm^*])$
22             $Pen \leftarrow Pen + P_{m^*+1} - P_{m^*} + P_{mm^*+1} - P_{mm^*}$
23             $M = \max\{M_1, \ldots, M_m\}$
24             if $(Pen, M)$ is dominated in $PE$ then
25                 $ContinuePhase1 \leftarrow False$
26             else
27                 $PE \leftarrow PE \bigcup \{(Pen, M)\}$
28         else
29             $ContinuePhase1 \leftarrow False$
30 else
31     $ContinuePhase1 \leftarrow False$
32 return S
```

distribution in the interval [1,50], the release dates and delivery times with an uniform distribution in the interval [1,K*20] with $K \in \{1, 10, 20, 30\}$. The value $K = 1$ is used in one instance and the three other values of K are used three times in the nine other instances. Each of these 20 instances is treated with 4 sets of penalties : $(P_2, P_3) \in \{(1, 2), (1, 1.5), (1, 2.5), (1, 5)\}$. For these instances we have that $J_3 \subset J_2 \subset J_1$ with $|J_1| = 20$, $14 \leq |J_2| \leq 16$ and $4 \leq |J_3| \leq 12$.

These specifications are indicated in the left part of the Tables 11 and 12. For all these instances with 20 jobs, it is possible to generate the exact Pareto front (PF) after a long enumeration. In the right part of these tables, we give the comparison between, respectively, the results of the first phase of the heuristic (Table 11) and the results of the heuristic with its two phases (Table 12), with the exact Pareto front.

The five columns of these right parts indicate:

(1) The total number of solutions (P, M) generated by the heuristic (first phase or both phases) for the 10 instances.
(2) The total number of solutions (P, M) inside the exact Pareto front for the 10 instances.
(3) The percentage of solutions inside the Pareto front obtained by the heuristic.
(4) The percentage of solutions inside the Pareto front that dominate a solution found by the heuristic with the same value of penalty P but a smaller value of M.
(5) The maximal error in percentage on M for the solutions of the preceding column.

Algorithm 9: BackTracking 2

```
Input  : A partition of the jobs on the machines
Output: A new partition of the jobs on the machines by transferring one job from machine $m^*$ to machine $m^* + 2$
1  $ContinuePhase1 \leftarrow True$
2  $I = \{i \in CP_{m^*} \bigcap J_{m^*+2}\}$
3  if $I \neq \emptyset$ then
4      foreach $i \in I$ do
5          $(M^{(i)}_{m^*}, CP^{(i)}_{m^*}) \leftarrow Heuristic(I_{m^*} \setminus \{i\})$
6          $(M^{(i)}_{m^*+2}, CP^{(i)}_{m^*+2}) \leftarrow Heuristic(I_{m^*+2} \bigcup\{i\})$
7      Select the best job $i^* \in I$
8      Select the best job $i^* \in I$ with $i^* = \arg\min_{i \in I} \{\max(M^{(i)}_{m^*}, M^{(i)}_{m^*+2})\}$
9      Transfer job $i^*$ on machine $m^* + 2 : I_{m^*} \leftarrow I_{m^*} \setminus \{i^*\}, I_{m^*+2} \leftarrow I_{m^*+2} \bigcup\{i^*\}$
10     $PUSH(S, [NoBT, i^*, m^*])$
11     $PUSH(S, [NoBT, i^*, m^* + 1])$
12     $Pen \leftarrow Pen + P_{m^*+2} - P_{m^*}$
13     $M = \max\{M_1, \ldots, M_m\}$
14     if $(Pen, M)$ is dominated in $PE$ then
15         $ContinuePhase1 \leftarrow False$
16     else
17         $PE \leftarrow PE \bigcup\{(Pen, M)\}$
18 else
19     $ContinuePhase1 \leftarrow False$
20 return S
```

Table 8. Data of the illustration.

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

The analysis of Tables 11 and 12 furnishes the following information.

- The solutions obtained at the beginning of phase 1, i.e. when successive jobs are transferred from m_1 till m_2, are always equal to those of the exact Pareto front (see the seven first solutions of the illustration in Table 9). It changes from the first solution with $m^* = 2$. As explained for the illustration in Table 9, at this moment often it is more

Table 9. Iterations of the first phase of heuristic.

(P, M)	M_1, M_2, M_3	Action	i^*	m^*	Order on the jobs on the three machines
(0,124)	**124**, 0, 0	–	0	1	1-17-10-16-5-8-6-19-14-15-3-18-20-4-7-13-9-11-12-2 // //
(1,114)	**114**, 18, 0	–	16	1	1-17-10-5-8-6-19-14-15-3-18-20-4-7-13-9-11-12-2 // **16** //
(2,104)	**104**, 33, 0	–	19	1	1-17-10-5-8-6-14-15-3-18-20-4-7-13-9-11-12-2 // **19**-16 //
(3,94)	**94**, 43, 0	–	18	1	1-17-10-5-8-6-14-15-3-20-4-7-13-9-11-12-2 // 19-**18**-16 //
(4,85)	**85**, 46, 0	–	10	1	1-17-5-8-6-14-15-3-20-4-7-13-9-11-12-2 // **10**-19-18-16 //
(5,76)	**76**, 55, 0	–	15	1	1-17-5-8-6-14-3-20-4-7-13-9-11-12-2 // 10-19-**15**-18-16 //
(6,67)	**67**, 64, 0	BT2	14	1	1-17-5-8-6-3-20-4-7-13-9-11-12-2 // 10-19-**14**-15-18-16 //
(7,66)	65, **66**, 0	BT1	8	1	1-17-5-6-4-7-20-3-13-9-11-12-2 // 10-**8**-19-15-18-14-16 //
(8,65)	**65**, 56, 33	–	18	2	1-17-5-6-4-7-20-3-13-9-11-12-2 // 10-8-19-15-14-16 // **18**
(9,60)	58, **60**, 33	BT1	12	1	1-17-5-6-4-7-20-3-13-9-11-2 // 10-8-19-15-14-16-**12** // 18
(10,58)	**58**, 50, 39	–	19	2	1-17-5-6-4-7-20-3-13-9-11-2 // 10-8-15-14-16-12 // **19**-18
(11,54)	54, **54**, 39	BT1	13	1	1-17-5-6-4-7-20-3-9-11-2 // 10-8-15-14-**13**-16-12 // 19-18

Table 10. Illustration of some backtrackings of the second phase of heuristic.

(P, M)	M_1, M_2, M_3	Action	i^*	m^*	Order on the jobs on the three machines
Backtracking of type 1 from the solution (10,58) **58**-50-39 in Table 9					
		–	17	1	
(12,51)	**51**, 50, 39	–	17	2	1-5-6-4-7-20-3-13-9-11-2 // 10-8-15-14-16-12 // **17**-19-18
Backtracking of type 1 from the solution (6,67) **67**-64-0 in Table 9					
		–	12	1	
(8,60)	**60**, 58, 33	–	18	2	1-17-5-8-6-3-20-4-7-13-9-11-2 // 10-16-19-14-15-**12** // **18**
Continue first phase					
(9,59)	59, **59**, 33	BT1	9	1	1-17-5-8-6-3-20-4-7-13-11-2 // 10-16-19-14-15-**9**-12 // 18
Backtracking of type 1 from the solution (8,60) **60**-58-33 above					
		–	17	1	
(10,54)	53, **54**, 39	–	19	2	1-5-8-6-3-20-4-7-13-9-11-2 // **17**-10-16-14-15-12 // **19**-18
Continue first phase					
(11,53)	**53**, 48, 39	–	17	2	1-5-8-6-3-20-4-7-13-9-11-2 // 10-16-14-15-12 // **17**-19-18
(12,51)	**51**, 50, 39	BT2	7	1	1-5-8-6-3-20-4-13-9-11-2 // 10-16-**7**-14-15-12 // 17-19-18
(13,51)	50, **51**, 39	BT1	9	1	1-5-8-6-3-20-4-13-11-2 // 10-16-7-14-15-**9**-12 // 17-19-18
Backtracking of type 2 from the solution (11,53) **53**-48-39 above					
		–	20	1	
(13,50)	**50**, 48, 39	–	20	2	1-5-8-6-3-4-7-13-9-11-2 // 10-16-14-15-12 // 17-19-18-**20**
Continue first phase					
(14,49)	49, **49**, 42	BT1	9	1	1-5-8-6-3-4-7-13-11-2 // 10-16-14-15-**9**-12 // 17-19-18-20

interesting that some jobs placed on m_2 at some preceding iterations go back to m_1 and are replaced by other jobs. The very high combinatorial aspect of such mechanism is not taken into account in the heuristic.

- Nevertheless, the two types of backtracking BT1 and BT2 avoid in part this difficulty. Even if the phase 1 generates already 50% of the solutions of the Pareto front, the phase 2 appears very complementary. Effectively the number of solutions equal to those of the Pareto front strongly increases with the phase 2. Moreover the number of dominated solutions with the same value of penalty strongly decreases and the maximal error made on M is reduced.
- We recall that initially the heuristic was designed for the case $P_2 = 1$ and $P_3 = 2$. Nevertheless, it appears that the results are relatively stable with different values for these penalties, even if the instances with $P_3 > 2\ P_2$ appear a little bit more difficult.

Table 11. Results on instances of 20 jobs and 3 machines – first phase.

Data	Instances	n	m	P_2	P_3	Phase 1	PF	Equals (%)	Dom. on M (%)	Error on M (%)
Set A	10	20	3	1	1.5	151	158	55	18	9
				1	2.0	151	158	55	28	8
				1	2.5	151	162	54	14	8
				1	5.0	151	162	54	14	8
Set B	10	20	3	1	1.5	141	160	50	19	10
				1	2.0	141	158	50	33	11
				1	2.5	141	174	46	20	11
				1	5.0	141	175	45	20	10

Table 12. Results on instances of 20 jobs and 3 machines – both phases.

Data	Instances	n	m	P_2	P_3	Both phases	PF	Equals (%)	Dom. on M (%)	Error on M (%)
Set A	10	20	3	1	1.5	156	158	87	7	3
				1	2.0	156	158	87	8	3
				1	2.5	167	162	86	8	3
				1	5.0	167	162	86	8	3
Set B	10	20	3	1	1.5	147	160	77	17	4
				1	2.0	145	158	77	18	4
				1	2.5	160	174	72	21	4
				1	5.0	164	175	71	20	4

- The results obtained for instances of set B are a little bit worse than those obtained with the set A. The main reason is that, in this second set, the processing times are randomly generated in a larger interval than in the first set. This larger dispersion of the processing times of the jobs increases the number of possibilities to replace a job on one machine by several other to reduce the final date. As said before, due to such combinatorial aspects, these instances are then more difficult to solve by the heuristic.
- In conclusion, the heuristic obtains a good approximation of a very large part of the exact Pareto front and its CPU time is always less than one second.

5.2 *Instances with 20 jobs and 4 machines*

The same two sets of instances (noted set A and set B) are considered except there are now four machines and thus four subsets are defined for the 20 jobs : $J_4 \subset J_3 \subset J_2 \subset J_1$ with $|J_1| = 20$, $15 \leq |J_2| \leq 17$, $11 \leq |J_3| \leq 13$ and $5 \leq |J_3| \leq 7$.

These specifications of the instances are indicated in the left part of the Tables 13 and 14. The right part of these tables presents the results obtained in a similar way than in the previous section.

The analysis of Tables 13 and 14 furnishes the following information.

- It appears first that the phase 1 is less efficient with 4 machines. The reason is that the difficulty described in the previous section is repeated now two times, not only for jobs transferred from m_1 till m_2, but also for jobs transferred from m_2 till m_3. Nevertheless, the phase 2 with its two types of backtracking remedies in large part this difficulty as shown by the results obtained.
- We note that the number of solutions obtained by the heuristic and in the exact Pareto front increases in the case of 4 machines. The reason is that there are more possibilities to distribute the jobs on the different machines. But it is important to underline that the global performance of the heuristic is not reduced by the consideration of this case.

5.3 *Instances with 30 jobs and 3 machines*

We randomly generate two sets (noted set A and set B) of 10 instances with 30 jobs and 3 machines with the penalties $P_1 = 0$, $P_2 = 1$ and $P_3 = 2$. In the first, the processing times are generated with an uniform distribution in the interval

[1,10], the release dates and delivery times with an uniform distribution in the interval [1,30]. In the second, based on Carlier (1982), the 10 instances are generated in the same way as the previews one of 20 jobs. For these instances we have that $J_3 \subset J_2 \subset J_1$ with $|J_1| = 30$, $18 \le |J_2| \le 26$ and $10 \le |J_3| \le 18$.

Table 13. Results on instances of 20 jobs and 4 machines – first phase.

Data	Instances	n	m	P_2	P_3	P_4	Phase 1	PF	Equals (%)	Dom. on M (%)	Error on M (%)
Set A	10	20	4	1	2	3	171	176	49	34	8
				1	3	6	171	182	47	22	8
				1	5	13	171	183	47	16	8
				5	8	10	171	177	49	19	11
Set B	10	20	4	1	2	3	180	198	39	41	12
				1	3	6	180	216	36	27	11
				1	5	13	180	224	34	16	7
				5	8	10	180	206	39	20	12

Table 14. Results on instances of 20 jobs and 4 machines – both phases.

Data	Instances	n	m	P_2	P_3	P_4	Both phases	PF	Equals (%)	Dom. on M (%)	Error on M (%)
Set A	10	20	4	1	2	3	177	176	88	9	3
				1	3	6	189	182	86	10	3
				1	5	13	191	183	85	10	3
				5	8	10	177	177	86	9	3
Set B	10	20	4	1	2	3	194	198	77	19	5
				1	3	6	215	216	74	21	5
				1	5	13	223	224	70	23	4
				5	8	10	201	206	77	17	5

Table 15. Results on instances of 30 jobs and 3 machines – first phase.

Data	Instances	n	m	P_2	P_3	Phase 1	PF	Equals (%)	Dom. on M (%)	Error on M (%)
Set A	10	30	3	1	2	224	233	56	30	10
Set B	10	30	3	1	2	245	264	46	40	11

Table 16. Results on instances of 30 jobs and 3 machines – both phases.

Data	Instances	n	m	P_2	P_3	Both phases	PF	Equals (%)	Dom. on M (%)	Error on M (%)
Set A	10	30	3	1	2	234	233	88	9	2
Set B	10	30	3	1	2	263	264	81	17	2

The analysis of Tables 15 and 16 furnishes the following information.

- We are still able to determine the exact Pareto front for such larger instances but the enumeration of the solutions take 3 hours, when the CPU time of the heuristic remains of 1 second.
- It is remarkable that the performance of the heuristic with its two phases is maintained, and the results are even a little bit better that those for instances with 20 jobs (see Table 16 by comparison with the line $P_2 = 1, P_3 = 2$ of Table 12).
- Clearly the number of solutions obtained by the heuristic and in the Pareto front increases with the number of jobs.

5.4 *Instances with 50 jobs and 3 machines*

Finally, we randomly generate also two sets (noted set A and set B) of 10 instances with 50 jobs and 3 machines with the penalties $P_1 = 0$, $P_2 = 1$ and $P_3 = 2$. In the first, the processing times are generated with an uniform distribution in the interval [1,10], the release dates and delivery times with an uniform distribution in the interval [1,50]. In the second, based on Carlier (1982), the 10 instances are generated in the same way as the previous one of 20 jobs. For these instances we have that $J_3 \subset J_2 \subset J_1$ with $|J_1| = 50$, $24 \leq |J_2| \leq 39$ and $14 \leq |J_3| \leq 27$.

The application of the heuristic is not limited by the number of jobs or the number of machines, but unfortunately, for large instances, it is no more possible to generate the exact Pareto front in a reasonable computing time. So for these present instances, the comparison can only be done between the results of phase 1 and those of the complete heuristic (in the second column of the right part of the table).

Table 17. Results on instances of 50 jobs and 3 machines – first phase.

Data	Instances	n	m	P_2	P_3	Phase 1	Heuristic	Equals (%)	Dom. on M (%)	Error on M (%)
Set A	10	50	3	1	2	278	336	61	18	8
Set B	10	50	3	1	2	348	362	51	27	10

The analysis of Table 17 furnishes the following information.

- We can observe that the percentage of solutions of phase 1 which are non-dominated by those of the complete heuristic are of 61 and 51%, respectively, for the two sets of instances (this is comparable with the results of the previous instances).
- For such larger instances and thus a larger number of solutions obtained by the heuristic, the CPU time of the heuristic is of 2 s.

6. 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 analyse the behaviour 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 7 where H_0^+ does not give the best value. Another related question to analyse 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 and illustrated for the instance of Section 4.3, the phase 2 of the present heuristic consider to replace on machine m^* only the last job transferred on machine $m^* + 1$ (see lines 7 till 9 of Algorithm 7) but it does not allow to replace on machine m^* a job placed at any preceding iteration on machine $m^* + 1$ (see the comment on the illustration of Section 4.3.). It is thus necessary to analyse if such possibility can be introduced inside the proposed method, of course with an increased amount of computations. A complete different way to heuristically tackle the problem may be also to propose a particular multi-objective metaheuristic adapted to the model (similarly to Ciavotta, Meloni, and Pranzo 2016), which solve the identical parallel machine scheduling problem, or to be adapted from Lin and Ying (2015) and Lin et al. (2016), as they solve the problem for unrelated parallel machines (see also Loukil, Teghem, and Tuyttens 2005).
- In the initial version of the paper, we only consider the case of three machines with penalties $P_2 = 1$ and $P_3 = 2$. So now, after the revision of our study, the present model is really larger as it considers any number of machines and any value of its penalties, and is applicable with anay number of jobs.
  Another interesting way to still extend the model is to introduce several machines at each level k, modifying the model of Section 3 into the model studied by Gharbi and Haouari (2002, 2007). It will be necessary to first examine

if some mechanisms of the four heuristics and the improvement algorithm of Section 3 can be generalised to this case.

Secondly, to examine if the bi-objective algorithm of Section 4 is adaptable to such more complex model can be of course a next step of our work. AQ11

Disclosure statement

No potential conflict of interest was reported by the authors. AQ12

References

Berrichi, A., and F. Yalaoui. 2013. “Efficient Bi-objective Ant Colony Approach to Minimize Total Tardiness and System Unavailability for a Parallel Machine Scheduling Problem.” *International Journal of Advanced Manufacturing Technology* 68 (9–12): 2295–2310.

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

Carlier, J. 1982. “The One Machine Sequencing Problem.” *European Journal of Operational Research* 11: 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: 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): 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): 1243–125.

Ciavotta, M., C. Meloni, and M. Pranzo. 2016. “Speeding Up a Rollout Algorithm for Complex Parallel Machine Scheduling.” *International Journal of Production Research* 54 (16): 4993–5009.

Ehrgott, M., and X. Gandibleux. 2002. “Multiobjective Combinatorial and Applications.” In *Multiple Criteria Optimization. State of the art. Annotated Bibliography. Survey*, edited by M. Ehrgott and X. Gandibleux. Kluwer. AQ13

Garey, M. R., and D. S. Johnson. 1989. *Computers and Intractability. A Guide to the Theory of NP-completeness*. Freeman Ed. AQ14

Gharbi, R. L., and M. Haouari. 2002. “Minimizing Makespan on Parallel Machines Subject to Release Dates and Delivery Times.” *Journal of Scheduling* 5: 329–355.

Gharbi, R. L., and M. Haouari. 2007. “An Approximate Decomposition Algorithm for Scheduling on Parallel Machines with Heads and Tails.” *Computer & Operations Research* 34 (3): 868–883.

Graham, R. L., E. L. Lawler, J. K. Lenstra, and A. H. C. Rinnooy Kan. 1979. “Optimization and Approximation in Deterministic Sequencing and Scheduling Theory: A Survey.” *Annals of Discrete Mathematics* 5: 287–326.

Haouari, M., and A. Gharbi. 2003. “An Improved Max-flow-based Lower Bound for Minimizing Maximum Latenesson Identical Parallel Machines.” *Operations Research Letters* 31 (1): 49–52.

Horn, W. A. 1974. “Some Simple Scheduling Algorithms.” *Naval Research Logistics Quarterly* 21: 177–185.

Lee, K., J. Y. T. Leung, and M. Pinedo. 2011. “Scheduling Jobs with Equal Processing Times Subject to Machine Eligibility Constraints.” *Journal of Scheduling* 14 (1): 27–38.

Lei, D. 2009. “Multi-objective Production Scheduling: A Survey.” *International Journal of Advanced Manufacturing Technology* 43: 926–938.

Leung, J. Y. T., and C. L. Li. 2008. “Scheduling with Processing Set Restrictions: A Survey.” *International Journal of Production Economics* 116: 251–262.

Leung, J. Y. T., and C. L. Li. 2016. “Scheduling with Processing Set Restrictions: A Literature Update.” *International Journal of Production Economics* 175: 1–11.

Leung, J. Y. T., H. Li, and M. Pinedo. 2008. “Scheduling Orders on Either Dedicated or Flexible Machines in Parallel to Minimize Total Weighted Completion Time.” *Annals of Operational Research* 159: 107–123.

Li, S.. 2016. “Parallel Machine Scheduling with Nested Processing Set Restrictions and Job Delivery Times.” *Mathematical Problems in Engineering* 2016: 1–5. Article ID 3203728. doi:10.1155/2016/3203728.

Li, C. L., and K. Lee. 2016. “A Note on Scheduling Jobs with Equal Processing Times and Inclusive Processing Set Restrictions.” *The Journal of the Operational Research Society* 67 (1): 83–8.

Li, C. L., and Q. Y. Li. 2015. “Scheduling Jobs with Release Dates, Equal Processing Times, and Inclusive Processing Set Restrictions.” *Journal of the Operational Research Society* 66 (3): 516–523.

Liao, L.-W., and G.-J. Sheen. 2008. “Parallel Machine Scheduling with Machine Availability and Eligibility Constraints.” *European Journal of Operational Research* 184 (2): 458–467.

Lin, C.-H., and C.-J. Liao. 2008. “Minimizing Makespan on Parallel Machines with Machine Eligibility Restrictions.” *The Open Operational Research Journal* 2: 18–2.

Lin, S.-W., and K.-C. Ying. 2015. “A Multi-point Simulated Annealing Heuristic for Solving Multiple Objective Unrelated Parallel Machine Scheduling Problems.” *International Journal of Production Research* 53 (4): 1065–1076.

Lin, S. W., Z. J. Lee, K. C. Ying, and C. C. Lu. 2011. “Minimization of Maximum Lateness on Parallel Machines with Sequence-dependent Setup Times and Job Release Dates.” *Computers & Operations Research* 38 (5): 809–815.

Lin, S.-W., K.-C. Ying, W.-J. Wu, and Y.-I. Chiang. 2016. “Multi-objective Unrelated Parallel Machine Scheduling: A Tabu-enhanced Iterated Pareto Greedy Algorithm.” *International Journal of Production Research* 54 (4): 1110–1121.

Loukil, T., J. Teghem, and D. Tuyttens. 2005. “Solving Multi-objective Production Scheduling Using Metaheuristics.” *European Journal of Operational Research* 161: 42–61.

Mateo, M., J. Teghem, and D. Tuyttens. 2016. “An Algorithm for a Bi-objective Parallel Machine Problem with Eligibility, Release Dates and Delivery Times of the Jobs.” *IFAC-PapersOnLine* 49 (12): 1614–1619.

Moghaddam, A., F. Yalaoui, and L. Amodeo. 2015. “Efficient Meta-heuristics Based on Various Dominance Criteria for a Single Machine Bi-criteria Scheduling Problem with Rejection.” *Journal of Manufacturing Systems* 34: 12–22.

Mohri, S., T. Masuda, and H. Ishii. 1999. “Bi-criteria Scheduling Problem on Three Identical Parallel Machines.” *International Journal of Production Economics* 60 (1): 529–536.

Moradi, E., and M. Zandieh. 2010. “Minimizing the Makespan and the System Unavailability in Parallel Machine Scheduling Problem: A Similarity-based Genetic Algorithm.” *International Journal of Advanced Manufacturing Technology* 51 (5–8): 829–840.

Naderi-Beni, M., E. Ghobadian, S. Ebrahimnejad, and R. Tavakkoli-Moghaddam. 2014. “Fuzzy Bi-objective Formulation for a Parallel Machine Scheduling Problem with Machine Eligibility Restrictions and Sequence-dependent Setup Times.” *International Journal of Production Research* 52 (19): 5799–5822.

AQ15 Pinedo, M. L. 2002. *Scheduling Theory, Algorithms, and Systems*. Prentice-Hall.

Recalde, D., C. Rutten, P. Schuurman, and T. Vredeveld. 2010. “Local Search Performance Guarantees for Restricted Related Parallel Machine Scheduling.” In *LATIN 2010*. Vol. 6034, LNCS, edited by A. López-Ortiz, 108–119. Heidelberg: Springer.

Shabtay, D., S. Karhi, and D. Oron. 2015. “Multipurpose Machine Scheduling with Rejection and Identical Job Processing Times.” *Journal of Scheduling* 18 (1): 75–88.

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

Tseng, C.-T., C.-H. Lee, Y.-S. P. Chiu, and W.-T. Lu. 2017. “A Discrete Electromagnetism-like Mechanism for Parallel Machine Scheduling Under a Grade of Service Provision.” *International Journal of Production Research* 55 (11): 3149–3163.

Wang, I. L., T. Yang, and Y. B. Chang. 2012. “Scheduling Two-stage Hybrid Flow Shops with Parallel Batch, Release Time, and Machine Eligibility Constraints.” *Journal of Intelligent Manufacturing* 23 (6): 2271–2280.

Ying, K. C., and H. M. Cheng. 2010. “Dynamic Parallel Machine Scheduling with Sequence-dependent Setup Times Using an Iterated Greedy Heuristic.” *Expert systems with Applications* 37 (4): 2848–2852.

Zarandi, M. H. F., and V. Kayvanfar. 2015. “A Bi-objective Identical Parallel Machine Scheduling Problem with Controllable Processing Times: A Just-in-time Approach.” *International Journal of Advanced Manufacturing Technology* 77 (1–4): 545–563.