

# ENUMERATION OF PARETO OPTIMA FOR UNIFORM PARALLEL MACHINE SCHEDULING WITH TWO CRITERIA

K.BOUIBEDE-HOCINE

*Université François-Rabelais de Tours*

*Laboratoire d'Informatique, 64 avenue Jean Portalis, 37200 Tours, France*

karima.hocine@etu.univ-tours.fr

V.T'KINDT

*Université François-Rabelais de Tours*

*Laboratoire d'Informatique, 64 avenue Jean Portalis, 37200 Tours, France*

tkindt@univ-tours.fr

**Abstract** In this paper we consider the solution of an uniform parallel machine scheduling problem with two criteria. The aim is to minimise the makespan as well as the maximum lateness of jobs. We first propose a heuristic by progressive construction which tries to solve the problem without studying all possible solutions. An efficient branch-and-bound algorithm to compute a strict Pareto optimum is presented. In this method we introduce the preemptive feasibility test and energetic test which are used to prune nodes in the search tree. We have also integrated other methods in this branch-and-bound algorithm to reduce the search space.

**Keywords:** Multicriteria Scheduling - Pareto optima - Feasibility.

## 1. Introduction

The scheduling problem that we consider can be stated as follows. Consider  $n$  jobs to be scheduled without preemption on  $m$  uniform parallel machines, such that each machine has its own speed denoted by  $V_j$ ,  $\forall j = 1, \dots, m$ . Each job  $J_i$  has a release time  $r_i$  and a due date  $d_i$ . Job  $J_i$  can be performed by any machine  $M_j$ , thus requiring a processing time  $p_{ij}$  with  $p_{ij} = \frac{p_i}{V_j}$ ,  $\forall i = 1, \dots, n, \forall j = 1, \dots, m$ , where  $p_i$  is the absolute processing time of the job  $J_i$ . The bicriteria scheduling problem

concerns the simultaneous minimisation of the makespan  $C_{max}$  and the maximum lateness  $L_{max}$ . For a given schedule, each job  $J_i$  completing at time  $C_i$ , the two objectives can be computed as follows:

$$C_{max} = \max_{i=1..n} \{C_i\} \text{ and } L_{max} = \max_{i=1..n} \{C_i - d_i\}.$$

It is remarkable that the solution of a multicriteria scheduling problem leads to calculate one or more strict Pareto optima since there rarely exists a single solution that optimises all criteria [T'kindt and Billaut, 2002]. A solution  $s$  is a strict Pareto optimum iff there does not exist another solution  $s'$  such that  $C_{max}(s') \leq C_{max}(s)$  and  $L_{max}(s') \leq L_{max}(s)$  with at least one strict inequality. Accordingly, criteria vector  $[C_{max}(s); L_{max}(s)]$  is said to be strictly non dominated. Notice that often the enumeration of strict Pareto optima is restricted to the enumeration of one solution per strictly non dominated criteria vectors. We denote by  $E$  this restricted set of strict Pareto optima. In this paper we focus on the enumeration of set  $E$  and following the classic three-field notation of scheduling problems introduced in [Graham et al, 1979] and extended to multicriteria scheduling problems in [T'kindt and Billaut, 2002], this problem can be referred to as  $Q/r_i, d_i/C_{max}, L_{max}$ . The calculation of the strict Pareto optima for the problem  $Q/r_i, d_i/C_{max}, L_{max}$  is NP-hard.

Related problems have been tackled in the literature. When only criterion  $L_{max}$  is minimised, the  $Q/r_i/L_{max}$  is NP-hard. The problem  $P/r_i, q_i/C_{max}$  is NP-hard in the strong sens because it is a generalization of the problem  $1/r_i, q_i/C_{max}$  which is NP-hard in the strong sens [Garey and Johnson, 1979]. An exact method has been proposed in [Tercinet, 2004] to solve the problem  $P/r_i, q_i/C_{max}$ .

The remainder is organized as follows. In Section 2 we present the main lines of the enumeration of strict Pareto optima as conducted in this paper. This enumeration is based on the successive solution of decision problems, for which we propose heuristic and exact algorithms in the rest of the paper. A heuristic which tries to solve quickly the problem is proposed in Section 3, whilst infeasibility tests which are based on the relaxation of the initial problem are introduced in Section 4. Section 5 is devoted to the presentation of a Branch and Bound method which uses the  $\epsilon$ -constraint and feasibility tests is presented in Section 6. In Section 7 we present computational testing.

## 2. Enumeration of the strict Pareto optima

To enumerate the strict Pareto optima a classic approach consists in iteratively solving  $\epsilon$ -constrained problems ([T'kindt and Billaut, 2002]).

The  $\epsilon$ -constrained problem considered in this paper is the following: we minimize the makespan subject to the additional constraint  $L_{max} \leq \epsilon$  where  $\epsilon$  is any given value. The corresponding scheduling problem is referred to as  $Q/r_i, d_i/\epsilon(C_{max}/L_{max})$ . It can be rewritten as:

$$(P_1) \quad Q/r_i, \tilde{d}'_i = d_i + \epsilon/C_{max}$$

Generally speaking, to each strict Pareto optimum  $s$  there exists an  $\epsilon$  value such that  $s$  is an optimal solution of the  $Q/r_i, d_i/\epsilon(C_{max}/L_{max})$  problem. The enumeration algorithm proceeds as follows. The value of  $\epsilon$  chosen at the first iteration is equal to a high value and let us denote by  $s_1$  the solution of the corresponding problem  $(P_1)$ . At the second iteration we set  $\epsilon = L_{max}(s_1) - 1$ . Again, problem  $(P_1)$  is solved and a new value  $\epsilon$  is set starting from the calculated solution. This process is iterated until problem  $(P_1)$  has no solution. To solve problem  $(P_1)$ , we can iteratively solve several problems:

$$(P_2) \quad Q/r_i, \tilde{d}'_i, C_{max} \leq \delta /-, \text{ where } \delta \text{ is any given value.}$$

Problem  $(P_2)$  is equivalent to:

$$(P_3) \quad Q/r_i, \tilde{d}_i = \min \{ \tilde{d}'_i, \delta \} /-$$

Let  $UB_{C_{max}}$  and  $LB_{C_{max}}$  be respectively an upper bound and a lower bound of the optimal  $C_{max}$  value for the  $Q/r_i, \tilde{d}'_i/C_{max}$  problem. The value of  $\delta$  used in problem  $(P_3)$  is defined by a dichotomic search, i.e. at each iteration  $\delta = (UB_{C_{max}} + LB_{C_{max}})/2$ . If the corresponding problem  $(P_3)$  is feasible, we set  $UB_{C_{max}} = \delta$ . Otherwise  $LB_{C_{max}} = \delta$ . This process stops when  $UB_{C_{max}} - LB_{C_{max}} = 1$  and we have  $C_{max}^* = UB_{C_{max}}$ .

We have transformed the solution of the optimization problem  $Q/r_i, d_i/C_{max}, L_{max}$  into the solution of a decision problem. We now focus on the solution of this decision problem.

### 3. A heuristic by progressive construction

This heuristic tries to solve the problem without studying all possible solutions. It can be seen as an alternate application of the Shortest Release Time (SRT) and Earliest Deadline (ED) rules. The SRT is applied, i.e. the job with the smallest value of  $r_i$  is scheduled, until the job with the smallest deadline must be scheduled otherwise, it would be infeasible if applying once more the SRT rule. A job is always scheduled on the machine on which it completes at the earlier. This algorithm requires

$o(n \log(n) + nm)$  time. Applying this heuristic enables to quickly detect if the problem is feasible, thus limiting the use of the exact method described in Section 6. If at the end of this heuristic no solution has been found, we must continue the solution of the feasibility problem by using an exact method. The algorithm of this heuristic is presented in Figure 1.

#### 4. Infeasibility tests

To study the infeasibility of the  $Q/r_i, \tilde{d}_i/-$  problem we can use two kinds of infeasibility tests which constitute sufficient conditions for deciding on the infeasibility of an instance, i.e. whenever they conclude to the infeasibility, the corresponding instance of the  $Q/r_i, \tilde{d}_i/-$  is infeasible. The first test, namely the preemptive approach, consists in enabling preemption of jobs. This relaxed decision problem can be polynomially solved by a network flow algorithm (see for instance [Brucker, 1995]). The second one, based on the energetic reasoning, has been first introduced in [Erschler et al, 1991] and [Lopez et al, 1992] and extended recently to uniform parallel machines in [T'kindt and Néron, 2004]. It is based on decomposing the scheduling horizon into time periods in which a mandatory part for each job, the energy requirement, has to be scheduled. If for a given time interval the sum of the energy requirements is greater than the processing capacities of the resources then the problem is not feasible. The energy requirement of job  $J_i$  in the interval  $[t_1, t_2]$  when assigned to machine  $M_j$ , and denoted by  $W(i, j, t_1, t_2)$  is defined as follows:

$$W(i, j, t_1, t_2) = \begin{cases} \min \left( t_2 - t_1; \frac{p_i}{V_j}; \max \left( 0, r_i + \frac{p_i}{V_j} - t_1; \right); \max \left( 0; t_2 - \tilde{d}_i + \frac{p_i}{V_j} \right) \right) \\ \quad \text{If } \frac{p_i}{V_j} \leq r_i - \tilde{d}_i \\ +\infty \quad \text{Otherwise} \end{cases}$$

Then we are faced, for each time interval, with the problem of assigning jobs to machines to test if all the mandatory parts fit in this interval. The details of this test is given in [T'kindt and Néron, 2004].

#### 5. Insertion of a job in a fixed sequence

This method enables us to add, in polynomial time, a job  $J_i$  to a partial solution  $\sigma_j$  on a given machine  $M_j$  or to decide that this insertion is not possible. It is used in the exact method provided in Section 6. Let us denote by  $t_k$  the starting time of job  $J_k$  in the partial schedule  $\sigma_j$ . We

```

/*L1 is the unscheduled job with the smallest ri*/
/*L2 is the unscheduled job with the smallest d̃i*/
While the list of unscheduled jobs is not empty
  If L2 can be scheduled on the machine with the smallest speed without infeasibility
  Then //Job L2 is not critical, so we try to assign the job L1
    We search for a machine on which job L1 completes without exceeding d̃L1
    If a machine has been found
    Then the job is assigned on this machine and it is deleted from
    the set of the remaining jobs to schedule
    Else End: The heuristic has not found a solution
    EndIf
  Else //Job L2 is critical, we try to find a machine which accept it
    We search for the first machine which accept job L2
    If a machine is found
    Then the job is assigned to this machine and it is deleted from
    the set of the remaining jobs to schedule
    Else End: The heuristic has not found a solution
    EndIf
  EndIf
EndWhile
End: The heuristic has found a solution

```

Figure 1. A heuristic for the solution of the  $Q/r_i, \tilde{d}_i/-$  problem

first identify the different blocks of  $\sigma_j$  and the idle times between these blocks. A block is a set of jobs in which the end of each job coincides with the start of the next job, i.e. if  $J_k$  precedes immediately  $J_l$  then  $C_k = t_l$  (Figure 2). Next the method determines timeshifts that can be done on blocks in order to enable the assignement of job  $J_i$ .

To determine if a job  $J_i$  can be scheduled in the idle period before a block  $B_p$ , we calculate the maximum right timeshift of this block. The timeshift is always made to the right, so blocks that precede the block on which we calculate the maximum timeshift do not intervene in this calculation. The schedule  $\sigma_j$  is a schedule "left timeshifted", so the first job  $J_k$  of each block starts at the date  $r_k$ . In this section we present a mathematical formulation of the maximum timeshift of a block, and we give the scheme of the insertion method. We denote by  $b$  the number of blocks. Let  $B_p$  be the block number  $p$ ,  $\pi_p$  the duration of the idle time after block  $B_p$ , and  $\delta_p^{max}$  the maximum timeshift which the block can have without considering others blocks. We have  $\delta_p^{max} = \min_{k \in B_p} (\tilde{d}_k - C_k)$ . The maximum timeshift of block  $B_p$  taking account of the following blocks in  $\sigma_j$  is denoted by  $\delta_{max}$ .

**Property:** The maximum timeshift  $\delta max$  of a block  $B_p$  is defined by:

$$\delta max = \sum_{u=p}^{b-1} \pi_u + \min_{1 \leq u \leq b-1} \left( \delta_u^{max} - \sum_{v=u}^{b-1} \pi_v; \delta_b^{max} \right)$$

**Proof:**

We suppose, without lost of generality, that  $p = 1$ .

- For  $b=1$ , we have one block, so  $\delta max = \delta_1^{max}$ .
- For  $b=2$ , we distinguish two cases:
  - 1 If  $\delta_1^{max} \leq \pi_1$ , then  $\delta max = \delta_1^{max}$ .
  - 2 If  $\delta_1^{max} > \pi_1$ , then  $\delta max = \pi_1 + \min(\delta_1^{max} - \pi_1; \delta_2^{max})$ .

We note that, if  $\delta_1^{max} \leq \pi_1$ , the value  $\delta_1^{max} - \pi_1$  is negative. We have also:

$$\forall u, \delta_u^{max} \geq 0 \text{ and } \min(\delta_1^{max} - \pi_1; \delta_2^{max}) = \delta_1^{max} - \pi_1.$$

So, if  $\delta_1^{max} \leq \pi_1$ ,  $\delta max = \pi_1 + \delta_1^{max} - \pi_1 = \delta_1^{max}$ . Therefore, we can deduce that:

$$\boxed{\delta max = \pi_1 + \min(\delta_1^{max} - \pi_1; \delta_2^{max})}$$

- For  $b=3$ , we distinguish three cases:
  - 1 If  $\delta_1^{max} \leq \pi_1$ , then  $\delta max = \delta_1^{max}$ .
  - 2 If  $\delta_1^{max} > \pi_1$  and  $\min(\delta_1^{max} - \pi_1; \delta_2^{max}) \leq \pi_2$ , we are in the case where  $b = 2$ , i.e,  $\delta max = \pi_1 + \min(\delta_1^{max} - \pi_1; \delta_2^{max})$ .
  - 3 If  $\delta_1^{max} > \pi_1$  and  $\min(\delta_1^{max} - \pi_1; \delta_2^{max}) > \pi_2$ , the third block can decrease the timeshift of the first block. So, we also have the case  $b = 2$ , and
 
$$\delta max = \pi_1 + \pi_2 + \min(\min(\delta_1^{max} - \pi_1; \delta_2^{max}) - \pi_2; \delta_3^{max})$$

We can deduce the mathematical formulation:

$$\boxed{\delta max = \pi_1 + \pi_2 + \min(\delta_1^{max} - \pi_1 - \pi_2; \delta_2^{max} - \pi_2; \delta_3^{max})}$$

- By analogy we deduce the following mathematical formulation:

$$\boxed{\delta max = \sum_{u=1}^{b-1} \pi_u + \min_{1 \leq u \leq b-1} \left( \delta_u^{max} - \sum_{v=u}^{b-1} \pi_v, \delta_b^{max} \right)}$$

After having identified the blocks and calculated for each of them their maximum right timeshift, the insertion method starts trying to insert job  $J_i$ . First it tries to assign it before the first block. If this job cannot be inserted, we shift the first block to the right, and we try again to assign this job before the first block. Next if this job cannot be assigned, we shift the second block always to the right and we try to schedule  $J_i$  between the first and the second blocks, and so on for all blocks  $p$ ,  $\forall p = 1, \dots, b$ . If job  $J_i$  cannot be inserted between two blocks, we try to place it after the last block. Clearly, job  $J_i$  cannot be scheduled such that it starts before  $r_i$  or completes after  $\tilde{d}_i$ . This method requires  $o(n)$  time.

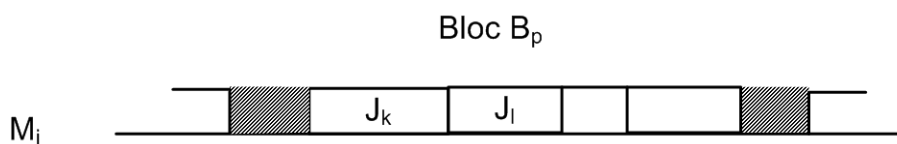


Figure 2. A block of the partial schedule  $\sigma_j$

## 6. A Branch and Bound procedure

The branch and bound method is widely used to solve scheduling problems. It is a solution procedure which systematically examines all possible combinations of sequencing jobs on machines. Before running the branch and bound method, the heuristic described in Section 3 is applied to try to decide on the feasibility of the problem. If it fails then the two infeasibility tests given in Section 4 are used to detect whether the instance is infeasible or not. If both feasibility and infeasibility have not been heuristically stated the branch and bound algorithm is run. This method calculates a feasible schedule, if it exists, by exploring a search tree in which each node is defined by, when applied to our problem, a partial schedule and the set of the remaining jobs to schedule. Starting from the root node, child nodes are created by assigning a job on the different machines. From the previous nodes, the method continues by selecting at each level a job and assign it on all machines. We continue in this way to create child nodes until we arrive to the leaf nodes. Let us consider a node for which we search to assign job  $J_i$  on machine  $M_j$ . We denote by  $\sigma_j$  the set of jobs assigned on the machine  $M_j$ , and  $\Omega$  the set of unscheduled jobs. We first heuristically try to detect if the job can be scheduled on the machine  $M_j$  in two steps. The first one consists

in inserting  $J_i$  in the sequence  $\sigma_j$  without introducing infeasibility (cf. Section 5), which can be achieved in  $o(n)$  time. If this insertion is not possible then the heuristic of Section 3 is applied on  $\sigma_j \cup \{J_i\}$  and to machine  $M_j$ . During one of these two steps, if we have constructed a feasible partial schedule then the current node is conserved. Otherwise we try to detect the infeasibility of this node by using the energetic test introduced in Section 4: if an assignment of a job on a machine is detected infeasible, the corresponding node is pruned. If the current node cannot be pruned by the infeasibility test, we have to exactly solve the  $1/r_i, \tilde{d}_i/-$  problem related to machine  $M_j$  with the branch and bound procedure of Carlier ([Carlier, 1982]). It is possible to get a solution from this algorithm, if this algorithm detects that the problem is feasible. This solution is kept as the new partial schedule of the child node. The current node is pruned if job  $J_i$  cannot be assigned to machine  $M_j$ .

## 7. Computational testings

Several computational experiences have been conducted in order to evaluate the performance of the heuristic by progressive construction and the exact method. Data sets which are used in the computational experiments are generated as follows: we generate problems with  $n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90\}$  and  $m \in \{2, 4, 6\}$ . To rule the distribution of due dates and release dates, two factors, namely  $L$  and  $R$  are used. Factor  $L$  is used to center the distribution of due dates and release dates while factor  $R$  rules their spreading. For each problem size  $(n; m)$ , different values of factor  $R$  are tested:  $R \in \{0.2; 0.4; 0.6; 0.8; 1.0; 1.2; 1.4; 1.6\}$ . The value of factor  $L$  in the uniform parallel machine problem is defined by  $L = \frac{\bar{P}}{\sqrt{V_{min}}}$  with  $V_{min} = \min_{j=1, \dots, m} V_j$ , and  $\bar{P} = \sum_{i=1}^n p_i$ . The machine speeds are drawn at random between 1 and 10 using an uniform distribution law. Absolute job processing times  $p_i$  are drawn at random between 10 and 100 using an uniform distribution law. For each job  $J_i$ , its due date  $d_i$  is drawn at random between  $(L - \frac{R \times \bar{P}}{2\bar{V}})$  and  $(L + \frac{R \times \bar{P}}{2\bar{V}})$ , with  $\bar{V} = \sum_{j=1}^m V_j$  using an uniform distribution law. Similarly, the release dates  $r_i$  are generated between  $[d_i - 3\frac{p_i}{V_{max}}; d_i - \frac{p_i}{V_{max}}]$ , with  $V_{max} = \max_{j=1, \dots, m} V_j$ . In case where  $r_i < 0$ , we set  $d_i = d_i - r_i$  and  $r_i = 0$ . For each tuple  $(n; m; R)$ , 30 instances are generated thus leading to the total of 240 instances per problem size. The experimental results show that the branch and bound algorithm is capable of solving problems with up to 30 jobs and 6 machines in a reasonable amount of time. More details about the computational results will be provided at the conference.

These results will also illustrate how it is hard to conduct the enumeration of strict Pareto optima.

## References

- Brucker P., Scheduling algorithms, *Springer, Heidelberg*, (1995).
- Lahrichi A., Ordonnancements: la notion de parties obligatoires et son application aux problèmes cumulatifs, *R.A.I.R.O.-R.O*,26:241–262, (1982).
- Carlier J., The one-machine sequencing problem, *European Journal of Operational Research*, 11:42–47, (1982).
- Erschler J., Lopez P., and Thuriot C., Raisonement temporel sous contraintes de ressources et problèmes d’ordonnement, *Revue d’Intelligence Artificielle*,5:7–32, (1991).
- Garey M.R. and Johnson D.S. Computers and intractability: a guide to the theory of NP-completeness. *W.H. Freeman and Company*, (1979).
- Graham R.L., Lawler E.L., Lenstra J.K., and Rinnooy Kan A.H.G. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*,5:287–326, (1979).
- Lopez P., Erschler J. and Esquirol P., Ordonnements de tâches sous contraintes: Une approche énergétique, *R.A.I.R.O.- A.P.I.I*,26:453–481, (1992).
- T’kindt V. and Billaut J-C. Multicriteria Scheduling: Theory, Models and Algorithms. *Springer (Heidelberg)*, (2002).
- T’kindt V. and Néron E., Energetic and preemptive feasibility tests for parallel machines scheduling, *Research report, Laboratoire Informatique, Université François-Rabelais de Tours*, 1–15, (2004).
- Tercinet F., Méthodes arborescentes pour la résolution des problèmes d’ordonnement, conception d’un outil d’aide au développement, *Ph.D.Thesis, Laboratoire Informatique, Université François-Rabelais de Tours*, (2004).