A Decomposition Approach to Characterize Feasible Schedules for the One-Machine Problem [1]

M-L. Levy[(1)] P. Lopez[(1)] B. Pradin[(1,2)]

[(1)]LAAS-CNRS – 7, av. du Colonel Roche – 31077 Toulouse Cedex – France
[(2)]INSA – Complexe Scientifique de Rangueil – 31077 Toulouse Cedex – France
emails : {levy,lopez,pradin}@laas.fr – phone : (33) 61 33 62 98 – fax : (33) 61 55 35 77

Abstract

This paper presents a decomposition approach for the one-machine scheduling problem with ready times, due dates and fixed processing times. We are not concerned here with searching for the best solution with regard to some criteria which is supposed to be relevant. The principles used rely on the constraint analysis aiming to characterize the set of feasible schedules. The objective is to approach this NP-complete problem by decomposing the initial problem into several subproblems. A temporal decomposition is used by means of the *rank intervals* which represent the feasible positions of each task in a sequence.

Content areas : Single-machine scheduling, Decomposition, Feasibility

Introduction

Generally the approaches addressing the scheduling issue envisage it as an optimization problem and propose exact or heuristic techniques aiming to find the best solution with regard to a given criterion (see for example [2] and [4][11][12] as surveys for the general scheduling problem, [3] and [14] as references for the one-machine scheduling problem). Since it might be ambitious to only reduce the knowledge on objectives and the quality of the solutions through use of a single criterion, others approaches focus on combining both analytic and domain-specific knowledges. Expert systems and decision-aid systems are then developed when humans and scheduling software systems have to co-operate [13].

Our work is not directly concerned with any solving procedure but with preparing for the phase of solution generation. This paper highlights the key role of the constraints ; indeed they implicitly define the feasibility of the solutions. We take up the approach, referred to as the *constraint analysis*, to define properties to be necessarily satisfied by all feasible schedules.

[1]To appear in: Proc. of International Conference on Industrial Engineering and Production Management (IEPM'95), Marrakech (Morocco), April 1955.

To face the complexity of large-scale problems, decomposition methods have been developed, in optimization [5][6][9][17][18], as well as in the decision-aid issue [1]. The principle is to transform the initial problem into several small-scale subproblems.

The purpose of this paper is to propose a decomposition method in order to characterize and to represent the set of feasible schedules for the one-machine scheduling problem, with regard to the constraint analysis objectives. The complete characterization of feasible schedules has been shown to be NP-complete [10]. To avoid the exponential complexity of searching for necessary and sufficient conditions of feasibility [7], we attempt to highlight only necessary conditions.

The problem is formulated as follows. A set T of n tasks is to be carried out on a single machine. Each task i of T is characterized by its ready time r_i, its due date d_i and a processing time p_i. The relation $d_i - r_i \geq p_i$ holds $\forall\, i = 1..n$. A task cannot be splitted. Two tasks cannot be carried out simultaneously on the machine. Let s_i be the start time of i. A schedule is *feasible* if it is compatible with the constraints of machine utilization as well as those of limit times, i.e., $s_i \geq r_i$ and $s_i + p_i \leq d_i$, $\forall\, i = 1..n$.

Basic operations for our approach are introduced in Section 1. Section 2 presents the method itself, examples and evaluation. Concluding remarks are then given.

1 Basic operations

Our decomposition approach is based on the determination of the feasible locations of each task in a sequence, defined as the rank intervals. The basic operations used later in our method to compute and update rank intervals, to update limit times and to decompose the set of tasks will be explained.

1.1 Slack and rank

Let m_{ij} be the slack time when task i is sequenced before task j, $i \neq j$. It can be written as : $m_{ij} = d_j - r_i - p_i - p_j$. Four cases are distinguished : (1) if $m_{ij} < 0$ and $m_{ji} < 0$, the problem is inconsistent ; (2) if $m_{ij} < 0$ and $m_{ji} \geq 0$ then i must be after j ; (3) if $m_{ij} \geq 0$ and $m_{ji} < 0$ then i must be before j ; (4) if $m_{ij} \geq 0$ and $m_{ji} \geq 0$ then i may be located before or after j. Let $Pre(k)$ (resp. $Post(k)$) be the set of tasks that necessarily precede (resp. succeed) task k. It yields :

$$\begin{cases} Pre(k) = \{i \ / \ i \neq k, \ m_{ki} < 0, \ m_{ik} \geq 0\}, \\ Post(k) = \{j \ / \ j \neq k, \ m_{kj} \geq 0, \ m_{jk} < 0\}. \end{cases}$$

For each task $k = 1..n$, the *rank interval* $[R_l(k), R_r(k)]$ represents the set of the feasible locations of k in the sequence. As the feasibility conditions pointed out are not sufficient, the rank intervals do not represent the set of the allowed locations but the set of the not forbidden ones. Thus :

$R_l(k) = |Pre(k)| + 1$ and $R_r(k) = n - |Post(k)|$.

The rank intervals are displayed in a chart that gives the tasks in the problem over the ranks (cf. Fig. 3).

1.2 Time and rank updating

Interacting all temporal constraints (limit times and precedence relations) may lead to an updating mechanism [8]. This can result in reducing time windows through basic constraint analysis rules. An updating is provided by using different rules, according to the type of the precedence relation : single, multiple and conjunctive, multiple and disjunctive precedences.

Single precedence
If i precedes j, then i must be completed before j can be initiated and j must be initiated after i can be completed. It follows :

$r_j \leftarrow \max(r_j, r_i + p_i)$ and $d_i \leftarrow \min(d_i, d_j - p_j)$.

Multiple conjunctive precedences
As a generalization, we now consider the case when a set of tasks B must precede a single task. Thus, $\forall\, b_i \in B$, $B \subseteq Pre(j)$, if b_i precedes j, then :

$r_j \leftarrow \max[r_j, eft(B)]$

where $eft(B)$ states for the earliest finishing time of set B. Let us notice that it is no need to build all possible sequences to compute $eft(B)$. It suffices to sort the tasks of B according to their increasing ready time and to evaluate a lower bound of their completion time.

Symmetrically, a due date updating can be computed ; $\forall\, b_j \in B'$, $B' \subseteq Post(i)$, if i precedes b_j, then :

$d_i \leftarrow \min[d_i, lst(B')]$

where $lst(B')$ states for the latest starting time of B'.

Multiple disjunctive precedences

Let $S \subset T$. The condition :

$$\max_{s \in S} d_s - \sum_{s \in S} p_s < r_j + p_j$$

leads to locating at least one task of S before j, yielding :

$$r_j \leftarrow \max[r_j, \min_{s \in S}(r_s + p_s)]$$

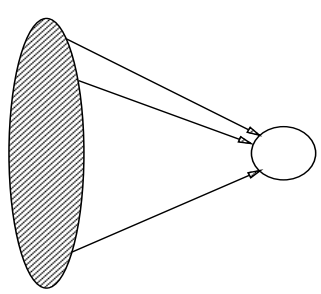

Fig. 1 - Disjunctive join

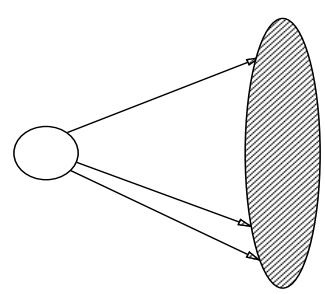

Fig. 2 - Disjunctive fork

i.e., j cannot start before the smallest $eft(s)$, $\forall\, s \in S$. This relation between S and j,

illustrated in Fig. 1, is referred to as a *disjunctive join*.

It may lead to updating $R_l(j)$ as well :

$R_l(j) \leftarrow \max[R_l(j), \min_{s \in S} R_l(s) + 1]$.

Symmetrically, we define a *disjunctive fork* when at least one task of $S' \subset T$ must succeed i (Fig. 2). It follows :

$$\min_{s \in S'} r_s + \sum_{s \in S'} p_s > d_i - p_i$$

and then $\begin{cases} d_i \leftarrow \min[d_i, \max_{s \in S'}(d_s - p_s)], \\ R_r(i) \leftarrow \min[R_r(i), \max_{s \in S'} R_r(s) - 1]. \end{cases}$

1.3 Regroupings

Three types of regroupings are distinguished : IRIS (Included Rank-Intervals Set), ERIS (Equal Rank-Intervals Set) and ORIS (Overlapped Rank-Intervals Set). The latter has been introduced in [1] as the so-called "semi-rigid subsequence". To each set $G \subset T$ is associated the rank interval :

$$\mathcal{R}(G) = [R_l(G), R_r(G)] \quad \text{with} \quad R_l(G) = \min_{k \in G} R_l(k) \quad \text{and} \quad R_r(G) = \max_{k \in G} R_r(k).$$

Definitions

Two tasks i and j belong to the same ORIS iff :

$\begin{cases} \mathcal{R}(i) \cap \mathcal{R}(j) \neq \emptyset \\ \textbf{or} \\ \exists G \subset T,\ i, j \notin G, \text{ such that :} \\ \mathcal{R}(i) \cap \mathcal{R}(G) \neq \emptyset \text{ and } \mathcal{R}(j) \cap \mathcal{R}(G) \neq \emptyset. \end{cases}$	i and j overlap one another or both i and j overlap a set of tasks G.

Two tasks i and j belong to the same ERIS iff :

$\mathcal{R}(i) = \mathcal{R}(j)$.	i and j have the same rank interval.

Two tasks i and j belong to the same IRIS iff :

$\begin{cases} \mathcal{R}(i) \subseteq \mathcal{R}(j) \text{ or } \mathcal{R}(j) \subseteq \mathcal{R}(i) \\ \textbf{or} \\ \exists G \subset T,\ i, j \notin G, \text{ such that :} \\ \mathcal{R}(i) \subset \mathcal{R}(G) \text{ and } \mathcal{R}(j) \subset \mathcal{R}(G). \end{cases}$	i and j have included rank intervals or there exists a set of tasks G whose rank interval includes $\mathcal{R}(i)$ and $\mathcal{R}(j)$.

Sequencing properties

- ORISs are minimum sets of tasks, independent with respect to sequencing, in all feasible solutions.

  According to the definition, tasks belonging to different ORISs do not share any rank. They are therefore sequenced independently from each other in all feasible solutions. When several tasks belong to the same ORIS, each of them overlaps at least another task in the ORIS. Thus, an ORIS cannot contain an independent subset of tasks.

- ERISs are sets of wholly permutable tasks.

  Tasks belonging to the same ERIS have the same rank interval, i.e. no precedence relation has been established between them. Their location in the sequence can thus be permuted in any way.

It must be noticed that the minimum-set property (for ORISs) and the permutability property (for ERISs) are linked with the feasibility conditions pointed out. Since the rank intervals do not represent allowed locations but not forbidden ones, they may be reduced by taking more feasibility conditions into account, modifying the decomposition in ORISs, ERISs and IRISs. It does not affect the independence properties already defined for the previous structure but may add new ones. It may affect permutability properties as well.

Hierarchical structure

This regouping types also constitute a hierarchical structure, with three levels : ERISs are subsets of IRISs and IRISs are subsets of ORISs, as illustrated in Fig. 3.

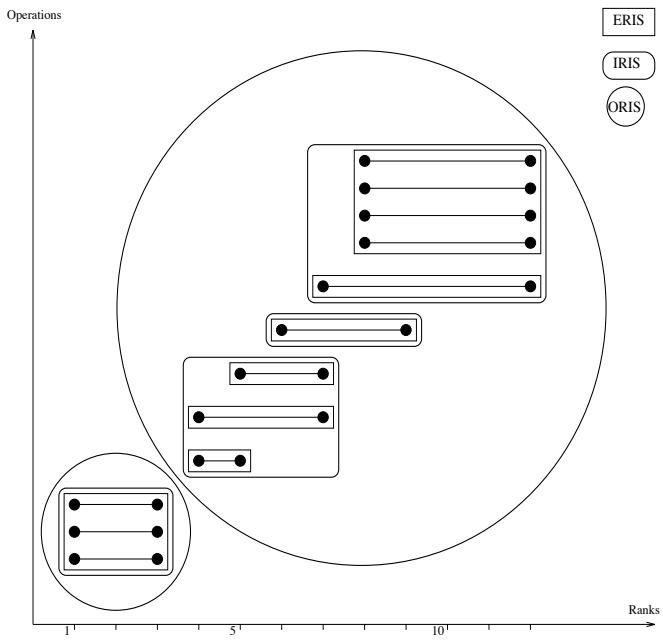

Fig. 3 - A decomposed set of tasks

The partition into ORISs is attractive as far as the problem is well structured enough so that independent subproblems can be pointed out (for most of the problems, only one ORIS is built, letting the problem not decomposed). ERISs often provide us with a too thin decomposition (i.e. small-size groups highly dependent from each other). To avoid this two drawbacks, we have chosen the IRISs, which offer a good tradeoff between set independence and task permutability inside a set.

2 Decomposition and constraint analysis

2.1 The algorithm

The method we present to characterize feasible schedules and to decompose the set of tasks consists in linked decomposition and updating procedures. Date updating leads to reducing rank intervals which may point out new groups ; this updating mechanism results from the determination of precedence relations, including conjunctive and disjunctive ones. As the algorithm to search for disjunctive precedences is of higher complexity, decomposition is used to prevent it from requiring too much computing time. Disjunctive joins and forks are therefore determined within each subproblem, whereas conjunctive precedences are searched over the whole problem.

Our method principles are summerized with the following three steps algorithm, repeated while ranks, groups or limit times are modified :

- The slack matrix is built. Its analyse allows each task i, $i = 1..n$, to be associated with $Pre(i)$ and $Post(i)$, expressing all conjunctive precedences. Dates are updated according to the constraint analysis rules and ranks are computed. The complexity of this step is $O(n^2)$.

- The set of tasks is decomposed into IRISs. This operation can easily be managed with the following procedure : assuming tasks are sorted according to their increasing left rank, we have to decide for each task i, $i = 1..n$, whether its rank interval allows it to belong to the last built group, or whether a new one must be created for i. A task may be able to belong to two different IRISs. In this case, it is assigned to the IRIS whose rank interval is the narrowest, or arbitrarily to one or the other if the rank intervals of the involved groups have the same width. This step runs with an $O(nlogn)$ computational complexity.

- Disjunctive precedences are searched within each group pointed out previously. Within each sub-problem, each task is attempted to be associated with a join (resp. a fork) and its ready time (resp. due date) is updated if the search has been successfull. The exhaustive search of disjunctive precedences has an exponential complexity $O(2^n)$. We have chosen to reduce again the computing time required by this operation with a partial search of disjunctive precedences. The complexity of the algorithm, explained in [15], is $O(n^3)$.

2.2 Example

Figs. 4 and 5 display the characteristics of the tasks to be scheduled. The decomposition/updating procedure perform in two steps. The first step leads to updating r_1, r_6, r_{11}, r_3, r_4, r_5, r_8 and d_2 by determining the sets Pre and $Post$ (Fig. 7). It follows some rank updatings for tasks $7, 11, 8, 3, 4$ and 5 (Fig. 6). Four IRISs (that are also ORISs) are found : $\{2, 9, 10\}, \{1, 6, 7, 12\}, \{11\}$ and $\{3, 4, 5, 8\}$. Join and fork relations allow r_2, r_9, d_{10}, d_6 and r_7 to be updated. During the second step, r_7 is increased again thanks to the slack matrix analysis. Figs. 8 and 9 display the final data. it can be noticed

that there is not much flexibility left : tasks 7, 10 and 11 are fixed ; tasks 2 and 9 are performed in sequence without any slack. The final results now point out six IRISs that are also ORISs.

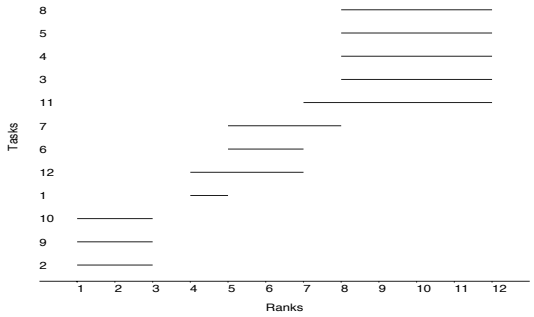

Fig. 4 - Initial problem (ranks)

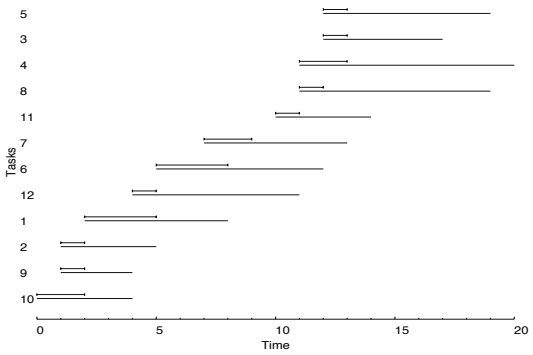

Fig. 5 - Initial problem (Gantt)

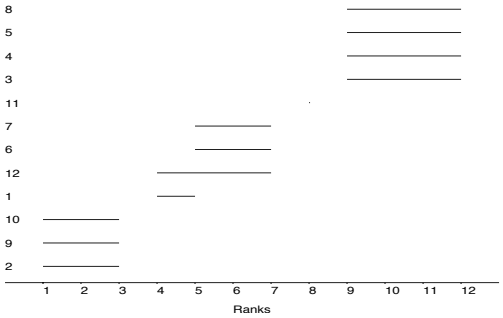

Fig. 6 - Intermediate ranks

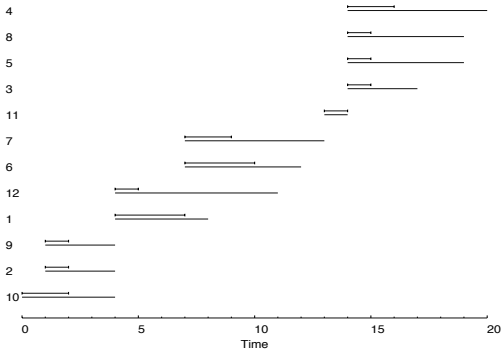

Fig. 7 - Intermediate Gantt

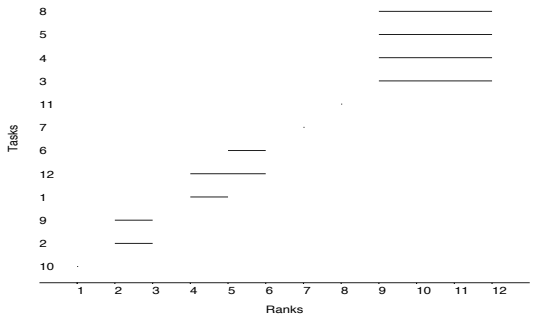

Fig. 8 - Updated ranks

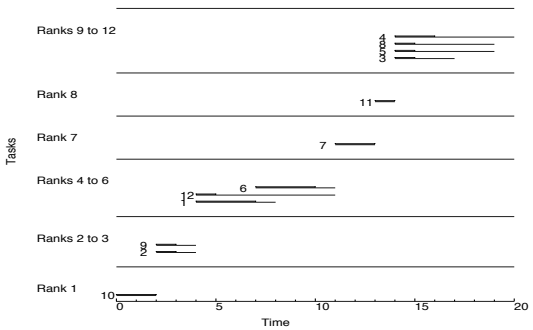

Fig. 9 - Final results

2.3 Method evaluation

Our method characterizes and represents feasible schedules for the one-machine problem : it determines temporal and sequential properties to be satisfied by all feasible schedules and decomposes the set of tasks according to their feasible locations in the sequence.

The only method concerned with the same goal we know so far is presented by Amamou et al. [1]. It uses sequential characterization based on conjunctive precedences to determine the rank intervals and it decomposes the set of tasks into "semi-rigid subsequences", i.e., ORISs.

We have improved this method in its both characterization and decomposition aspects, taking into account more constraint analysis rules (sequential and temporal characterization for conjunctive and disjunctive precedences) and using a lower decomposition level (the IRISs).

The ranks intervals computed by Amamou's method for the above example are displayed Fig. 10. No limit time are updated since the method focuses on sequential characterization. Two semi-rigid subsequences ({2, 9, 10} and {1, 3, 4, 5, 6, 7, 8, 11, 12}) are pointed out whereas six IRISs, that appear to be also ORISs, were raised from our iterative procedure. Moreover, our approach highligths more feasibility conditions, thanks to the limit times updating and to the disjunctive precedences.

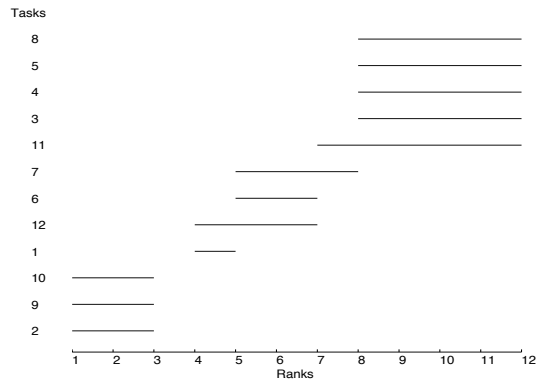

Fig. 10 - Amamou's method

Nevertheless, we do not determine all feasibility conditions that indeed hold, since our conditions are not sufficient. An algorithm proceeding in two steps is presented by Erschler et al. [7], to characterize feasible schedules with necessary and sufficient conditions. First of all, it uses the interdependency of temporal and sequential characterization in an iterative procedure. More complex conditions are then taken into account to determine the feasibility conditions not included in those already found. Our method would give the same results as the first step of the algorithm if we had not reduced the search of disjunctive precedences.

The capability of our method for the characterization stage is evaluated through the study of the following small-size example. The initial problem and the final results are presented respectively in Tables 1 and 2 ; r_u and d_u stand for the updated ready times and due dates. Only two sequences are feasible, *abcd* and *acbd*, as shown by the rank intervals. Our method has found the necessary and sufficient feasibility conditions pointed out in [7] ; the reason is that the first step of the algorithm suffices to characterize the feasible schedules of this example.

If the ready time of task a is modified, setting $r_a = 1$, our method computes the results given in Table 3, identical with those obtained after the first step of Erschler's algorithm. The second step of this algorithm would allow all feasibility conditions to be highlighted, with the sequencing relation "b before d".

tasks	a	b	c	d
r	2	5	10	9
d	18	24	23	27
p	5	7	7	3

Table 1 -

tasks	a	b	c	d
r_u	2	7	10	21
d_u	10	24	23	27
R_l	1	2	2	4
R_r	1	3	3	4

Table 2 -

tasks	a	b	c	d
r_u	1	6	10	13
d_u	10	24	23	27
R_l	1	2	2	2
R_r	1	4	4	4

Table 3 -

Our decomposition approach points out properties to be necessarily satisfied by all feasible schedules. It can therefore forget some conditions, as it is illustrated through this example. Furthermore, the complexity of our procedures remains polynomial, whereas the characterization of feasible schedules with necessary and sufficient conditions is exponential.

Concluding remarks

Our method structures the set of tasks according to their temporal properties. We expect such a method to be used in an interaction context, between humans and scheduling software systems. It could specially help a human operator in his decision-making by providing him with reduced set of tasks to schedule.

The method is not concerned with any solving procedure but with characterizing feasible schedules. We have compared our results with those given by the methods presented in [1] and [7], concerned with the same goal. As we wish to go further in the problem solving, we trend towards developing a heuristic technique based on rank intervals properties, to produce a feasible schedule as a solution of the problem.

An extension for multimachine problems has been considered as well [16]. The method used independently over each machine, assuming limit times are computed for every operation of each job, offers a good characterization of feasible schedules for the Flow-shop problem. The Job-shop problem is quite more difficult : the differences between the routings of jobs for each machine add feasibility conditions connected with machines interdependency.

References

[1] Amamou, M., Happiette, M. and Staroswiecki, M., *Decomposition of the single machine scheduling based on the notion of semi-rigid subsequences*, International Conference on Automation Technology, Taiwan, July 4-6, 1992.

[2] Baker, K., *Introduction to sequencing and scheduling*, New York, Wiley, 1974.

[3] Carlier, J., *The one-machine sequencing problem*, IJPR, **11**, 1982, pp.42-47.

[4] McCarthy, B.L. and Liu, J., *Addressing the gap in scheduling research : a review of optimization and heuristic methods in production scheduling*, IJPR, **31**, 1993, pp.59-79.

[5] Chambers, R., Carraway, R., Towe, T. and Morin, T., *Dominance and decomposition heuristics for single machine schduling*, Oper. Res., **39**, 4, 1991, pp.639-647.

[6] Chu, C., Portmann, M-C. and Proth, J-M., *A splitting-up approach to simplify job-shop scheduling problems*, IJPR, **30**, 1992, pp.859-870.

[7] Erschler, J., Roubellat, F. and Vernhes, J.P., *Characterizing the set of feasible sequences for n jobs to be carried out on a single machine*, EJOR, **4**, 1980, pp.189-194.

[8] Erschler, J. and Esquirol, P., *Decision-aid in job-shop scheduling : a knowledge-based approach*, IEEE International Conference on Robotics and Automation, San Antonio, CA, 1986, pp.1641-1656.

[9] Erschler, J. and Pradin, B., *Décomposition temporelle et problèmes d'ordonnancement*, Distancia'92, Rennes, France, June 22-26, 1992.

[10] Garey, M. and Johnson, D., *Computers and intractability : a guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.

[11] GOThA, *Les problèmes d'ordonnancement*, RAIRO, Rech. Op., **27**, 1993, pp.77-150.

[12] Graves, S., *A review of production scheduling*, Oper. Res., **29**, 4, 1981.

[13] Kusiak, A., *Knowledge-based systems in manufacturing*, Ed., Taylor & Francis, New-York, 1989.

[14] Lageweg, B., Lenstra, J. and Rinnooy Kan, A., *Minimizing maximum lateness on one machine : computational experience and some applications*, Statis. Neerlandica, **30**, 1976, pp.25-41.

[15] Levy, M-L., Lopez, P. and Pradin, B., *Décomposition temporelle pour la caractérisation d'ordonnancements admissibles*, LAAS technical report, **94171**, 1994.

[16] Levy, M-L., Lopez, P. and Pradin, B., *Characterization of feasible schedules for the flow-shop problem : a decomposition approach*, IMSE'94, Grenoble, France, December 12-14, 1994.

[17] Portmann, M-C., *Méthodes de décomposition spatiale et temporelle en ordonnancement de la production*, RAIRO-APII, **22**, 1988, pp.439-451.

[18] Potts, C. and Van Wassenhove, L., *Dynamic programming and decomposition approaches for the single machine total tardiness problem*, EJOR, **32**, 1987, pp.405-414.