Polynomial Time Algorithms for Minimizing the Weighted Number of Late Jobs on a Single Machine with Equal Processing Times

Philippe Baptiste

UMR CNRS 6599, HEUDIASYC,
Université de Technologie de Compiègne,
Centre de Recherches de Royallieu,
Rue Personne de Roberval, BP 20.529,
60205 Compiègne Cedex, France

e-mail: Philippe.Baptiste@hds.utc.fr
URL: http:\\www.hds.utc.fr\~baptiste

Polynomial Time Algorithms for Minimizing the Weighted Number of Late Jobs on a Single Machine with Equal Processing Times

Abstract

We study the problem of minimizing the weighted number of late jobs to be scheduled on a single machine when processing times are equal. In this paper, we show that this problem, as well as its preemptive variant, are both strongly polynomial. When preemption is not allowed $1|p_j=p,r_j|\Sigma w_j U_j$, the problem can be solved in $O(n^7)$. In the preemptive case, $1|p_j=p,pmtn,r_j|\Sigma w_j U_j$, the problem can be solved in $O(n^{10})$. Both algorithms are based upon dynamic programming.

Keywords

Single machine scheduling, preemptive scheduling, late jobs, dynamic programming.

1. Introduction

Given a set of jobs $P = \{J_1, ..., J_n\}$, each job being described by a release date r_i, a due date d_i, a processing time p_i (with $r_i+p_i \leq d_i$), and a non-negative weight w_i, minimizing the weighted number of late jobs on a single machine consists of finding a schedule of the jobs on which the weighted number of jobs completed after their due date is minimal. Conversely, one can also seek to maximize the weighted number of jobs completed before their due date. Along this paper, we will rather rely on this latter formulation. This problem, denoted as $1|r_j|\Sigma w_j U_j$ (where U stands for Unit penalty per late job) in the standard scheduling terminology (*e.g.*, [3]) is NP-hard in the strong sense, even if weights are equal [10]. To our knowledge, no exact approach has been made to solve the general problem. However, some branch and bound methods have been recently designed for the non-weighted problem ([2], [7], [9], [15]).

Some special cases of the general problem are solvable in polynomial time (see [4] for a rather comprehensive and up-to-date review of the hardness of machine scheduling). When weights are equal and when release dates are equal $1||\Sigma U_j$, the problem can be solved in $O(n\log(n))$ steps by Moore's well-known algorithm [14]. Moreover, when release and due dates of jobs are ordered similarly ($[r_i<r_j] \Rightarrow [d_i \leq d_j]$), the same problem is solvable in $O(n^2)$ with a dynamic programming algorithm of Kise, Ibaraki and Mine ([11]). An $O(n \log(n))$ has been proposed Lawler for the same problem ([13]). Dauzère-Pérès and Sevaux have extended [8] the result of Kise, Ibaraki and Mine to the

case where ($[r_i < r_j] \Rightarrow [d_i \leq d_j$ or $r_j+p_j+p_i>d_i]$). Carlier has proven [5, 6] that when processing times are equal, the problem $1|p_j=p,r_j|\Sigma U_j$ can be solved in $O(n^3 \log(n))$.

Researchers have also studied the preemptive variant of the problem $1|pmtn,r_j, \Sigma w_j U_j$. It is NP-hard but can be solved in pseudo-polynomial time by Lawler's algorithm [12, 3] whose time and space complexities are respectively $O(n\,k^2W^2)$ and $O(k^2W)$, where k is the number of distinct release dates and where W is the sum of the weights of the jobs. If weights are equal $1|pmtn,r_j|\Sigma U_j$, the problem obviously becomes strongly polynomial; the time and space bounds of Lawler's algorithm reducing to $O(n^3k^2)$ and $O(n\,k^2)$. So, $O(n^5)$ and $O(n^3)$ if all release dates are distinct. [1] describes an algorithm for this special case that improves the bounds to respectively $O(n^4)$ and $O(n^2)$.

These results leave two problems open, namely the non-preemptive and the preemptive weighted problems with equal processing times: $1|p_j=p,r_j|\Sigma w_j U_j$ and $1|p_j=p,pmtn,r_j|\Sigma w_j U_j$. In this paper, we show that both of them are strongly polynomial and we provide two dynamic programming algorithms whose time-bounds are respectively $O(n^7)$ and $O(n^{10})$. This paper is organized as follows. Sections 2 and 3 are dedicated respectively to the non-preemptive weighted problem $1|p_j=p,r_j|\Sigma w_j U_j$ and to the preemptive weighted problem $1|p_j=p,pmtn,r_j|\Sigma w_j U_j$. Finally, we draw some conclusions in Section 4.

2. The Non-Preemptive Weighted Problem ($1|p_j=p, r_j|\Sigma w_j U_j$)

A schedule of a subset X of jobs is said to be feasible *iff* (1) all jobs in X start after their release date and are completed after their due date and (2) jobs do not overlap in time. Notice that this definition remains valid in the preemptive case. Since when a job is late, it can be scheduled arbitrarily late, the problem reduces to finding a set of jobs (1) that is **feasible**, *i.e.*, for which there exists a schedule that meets release dates and due dates and (2) whose weight is maximal. From now on, we suppose that jobs are sorted in increasing order of due dates. We first introduce some notation and then provide the proposition that is the basis of our dynamic programming algorithm.

Definition.

Let $\Theta = \{t \text{ such that } \exists\, r_i, \exists\, l \in \{0, \dots, n\} \mid t = r_i + l * p\}$. □

Notice that there are at most n^2 values in Θ.

Proposition 1.

On any left-shifted schedule (*i.e.*, on any schedule on which jobs start either at their release date or immediately after another job), the starting times of jobs belong to Θ.

Proof.

Let J_k be any job. Let t be the largest time point before the start time of J_k at which the start-time of a job is equal to its release date (such a time-point exists because the schedule is left-shifted). Thus, t is a release date, say r_i. Between r_i

and the starting time of J_k, l jobs execute ($0 \le l \le$ n). Hence the starting time of J_k belongs to Θ. □

Since any schedule can be left-shifted, Proposition 1 induces a simple dominance property: There is an optimal schedule on which jobs start at time points in Θ.

Definition.

- For any integer $k \le n$, let $U_k(s,e)$ be the set of jobs whose index is lower than or equal to k and whose release date is in the interval $[s,e)$, *i.e.*, $U_k(s, e) = \{J_i \mid i \le k$ and $s \le r_i < e\}$.
- Let $W_k(s,e)$ be the maximal weight of a subset of $U_k(s,e)$ such that there is a feasible schedule S of these jobs such that
  - S is idle before time $s+p$,
  - S is idle after time e,
  - starting times of jobs on S belong to Θ.

  Notice that if the subset of $U_k(s,e)$ is empty, $W_k(s,e)$ is equal to 0. □

Proposition 2. (*cf.*, Figure 1)

For any value of k in $[1, n]$ and for any values s, e with $s \le e$, $W_k(s,e)$ is equal to $W_{k-1}(s,e)$ if $r_k \notin [s,e)$ and to the following expression otherwise:

$$\max(W_{k-1}(s,e), \max_{\max(r_k, s+p) \le s' \le \min(d_k, e)-p, s' \in \Theta} (w_k + W_{k-1}(s,s') + W_{k-1}(s',e))).$$

Proof (sketch).

Let W' be the expression above. If $r_k \notin [s,e)$, the result obviously holds since $U_k(s,e) = U_{k-1}(s,e)$. We now consider the case where $r_k \in [s,e)$.

We first prove that $W' \leq W_k(s,e)$.

- If $W' = W_{k-1}(s,e)$ then $W' = W_{k-1}(s,e) \leq W_k(s,e)$.
- If there is a value s' in Θ such that $\max(r_k, s+p) \leq s' \leq \min(d_k,e)-p$ and such that $W' = w_k + W_{k-1}(s,s') + W_{k-1}(s',e)$, let X and Y be two subsets of the set of jobs P that realize respectively $W_{k-1}(s,s')$ and $W_{k-1}(s',e)$. Because of the definition of W, the sets X and Y are disjoint and moreover, there exists a feasible schedule of X that fits in $[s+p,s']$ and there exists a feasible schedule of Y that fits in $[s'+p,e]$. Thus, $X \cup Y \cup \{J_k\}$ is a set whose weight is W' and there is a schedule of the jobs in this set that does not start before $s+p$ and that ends before e (take the schedule of X, schedule J_k at time s' and add the schedule of Y, see Figure 1). On top of that, starting times belong to Θ. Hence, $W' \leq W_k(s,e)$.

We now prove that $W_k(s,e) \leq W'$.

Let Z be a subset that realizes $W_k(s,e)$. If J_k does not belong to Z then $W_k(s, e) = W_{k-1}(s, e) \leq W'$. Now suppose that J_k belongs to Z. According to the definition of $W_k(s, e)$, there is a schedule S of Z that fits in $[s + p, e]$ on which starting times belong to Θ.

We claim that we can suppose that on S, the jobs executed after J_k are not available when J_k starts (*i.e.*, their release date is strictly greater than the start time of J_k). To justify our claim, we show how S can be modified to reach this

property: Suppose that there is a job J_i that starts after J_k and that is available at the start time of J_k. Let then $f(S)$ be the schedule obtained by exchanging the starting times of J_i and J_k. Because $d_i \leq d_k$ and because processing times are equal, the resulting schedule is feasible. Notice that each time f is applied, the position of J_k strictly increases and that the idle time intervals of the resource remain the same. Thus, f can be applied a limited number of times only. The resulting schedule is feasible and the jobs executed after J_k are not available at the starting time of J_k. On top of that the overall set of starting times has not been modified.

Let us examine the partition induced by the starting time s' of J_k.

- The jobs scheduled before s', belong to $U_{k-1}(s, s')$ and their total weight is lower than or equal to $W_{k-1}(s, s')$.
- The jobs scheduled after J_k belong to $U_{k-1}(s', e)$ and their total weight is lower than or equal to $W_{k-1}(s', e)$.
- The weight of J_k is w_k.

Moreover s' belongs to Θ because it is a starting time. On top of that, $\max(r_k, s + p) \leq s' \leq \min(d_k, e) - p$. Hence, the weight of the set Z is lower than or equal to W'. □

Given the dominance property induced by Proposition 1, the maximum weighted number of on-time jobs is $W_n(\min\Theta - p, \max\Theta)$. Thanks to Proposition 2, we have a straight dynamic programming algorithm to compute this value. The relevant values for s and e are exactly those in Θ (plus $\min\Theta - p$ for s). The values of $W_k(s, e)$ are stored in a multi-dimensional array of size

$O(n^5)$ (n possible values for k, n^2 possible values for s and n^2 possible values for e). Our algorithm then works as follows.

- In the initialization phase, $W_0(s, e)$ is set to 0 for any values s, e ($s \leq e$) in Θ.
- We then iterate from $k = 1$ to $k = n$. Each time, W_k is computed for all the possible values of the parameters thanks to the formula of Proposition 2 and to the values of W_{k-1} computed at the previous step.

The initialization phase can be done in $O(n^4)$. Afterwards, for each value of k, $O(n^4)$ values of W_k have to be computed. For each of them, a maximum among $O(n^2)$ terms is computed. This leads to an overall time complexity of $O(n^7)$. A rough analysis of the space complexity leads to an $O(n^5)$ bound but since, at each step of the outer loop on k, one only needs the values of W computed at the previous step (k-1), the algorithm can be implemented with 2 arrays of $O(n^4)$ size (one for the current values of W and one for the previous values of W).

3. The Preemptive Weighted Problem ($1|p_j=p, pmtn, r_j|\Sigma w_j U_j$)

Again, we suppose that jobs are sorted in increasing order of due dates. We first introduce Jackson Preemptive Schedule and some notation. Afterwards, we provide the proposition that is the basis of our dynamic programming algorithm.

Definition.

The Jackson Preemptive Schedule (JPS) of a set of jobs $O \subseteq P$ is the preemptive schedule obtained by applying the Earliest Due Date priority dispatching rule: whenever the machine is free and one job in O is available,

schedule the available job $J_i \in O$ for which d_i is the smallest. If a job J_j becomes available while J_i is in process, stop J_i and start J_j if $d_j < d_i$, otherwise continue J_i. □

Jackson Preemptive Schedule has several interesting properties (*e.g.*, [6]). In particular, if a job is scheduled on JPS after its due date, there is no feasible preemptive schedule of the set of jobs. Hence, searching for a schedule on which the weighted number of late jobs is minimal, reduces to finding a set whose weight is maximal and that is feasible, *i.e.*, whose JPS is feasible.

As for the non-preemptive case, the time points in Θ play a particular role in the structure of optimum schedules. In the following, we note $\Theta = \{t_1, t_2, ..., t_q\}$ the ordered set of distinct time-points in Θ. Recall that $q \leq n^2$.

Proposition 3.

For any subset of jobs Z, the start and end times of the jobs on the JPS of Z belong to the set Θ.

Proof.

We first prove that the end time of a job on the JPS of Z belongs to Θ. Let J_k be any job and let s and e be respectively its start and end times on JPS. Let t be the minimal time point such that between t and s JPS is never idle. Because of the structure of JPS, t is a release date, say r_x. The jobs that execute (even partially) between s and e execute neither before s nor after e (because Jackson Preemptive schedule is based upon the EDD rule). Thus $e - s$ is a multiple of p. Two cases can occur:

- Either J_k causes an interruption and hence $s = r_k$.
- Or J_k does not cause any interruption and hence the jobs that execute between r_x and s, are fully scheduled in this interval. Consequently, $s - t$ is a multiple of p.

In both cases, there is a release date r_y (either r_k or r_x) such that between r_y and e, JPS is never idle and such that e is equal to r_y modulo p. On top of that, the distance between r_y and e is not greater than $n * p$ (because JPS is not idle). Hence, $e \in \Theta$.

Now consider the start time of any job on JPS. This time point is either the release date of the job or is equal to the end time of the "previous" one. Thus, start times also belong to Θ. □

Definition.

For any time points t_u, t_v in Θ with $u < v$ and for any integer value k such that $1 \leq k \leq n$,

- let $U_k(t_u, t_v) = \{J_i \mid i \leq k \text{ and } t_u \leq r_i < t_v\}$ (as for the non-preemptive case),
- for any m such that $1 \leq m \leq n$, let $W_k(t_u, t_v, m)$ be the maximal weight of a subset $Z \subseteq U_k(t_u, t_v)$ of m jobs such that, the JPS of Z is feasible and ends before t_v. If there is no such subset, $W_k(t_u, t_v, m)$ is set to $-\infty$.

□

Proposition 4. (*cf*., Figure 2)

For any time points t_u, t_v in Θ with $u < v$ and any integer values k and m such that $1 < k \leq n$ and $1 \leq m \leq n$, $W_k(t_u, t_v, m)$ is equal to $W_{k-1}(t_u,t_v,m)$ if $r_k \notin [t_u,t_v)$ and to the expression above otherwise:

$\max(W_{k-1}(t_u,t_v,m),$

$$\max_{\substack{t_x,t_y\in\Theta,\\ \max(r_k,t_u)\leq t_x<t_y\leq\min(d_k,t_v)\\ m_1+m_2+m_3=m-1,\\ p^*(m_2+1)=t_y-t_x}} (W_{k-1}(t_u,t_x,m_1)+W_{k-1}(t_x,t_y,m_2)+W_{k-1}(t_y,t_v,m_3))+w_k)$$

Proof.

Let W' be the expression above. It is obvious that if $r_k \notin [t_u,t_v)$, $W_k(t_u, t_v, m)$ is equal to $W_{k-1}(t_u,t_v,m)$. In the following, we suppose that $r_k \in [t_u,t_v)$.

We first prove that $W' \leq W_k(t_u,t_v,m)$.

- Consider the case where $W'=W_{k-1}(t_u,t_v,m)$. Since $U_{k-1}(t_u,t_v)\subseteq U_k(t_u,t_v)$, we have $W' \leq W_k(t_u,t_v,m)$.
- Consider now the case where there exist $t_x\in\Theta, t_y\in\Theta$ and 3 integers m_1,m_2,m_3 such that
  - $\max(r_k,t_u)\leq t_x<t_y\leq\min(d_k,t_v)$,
  - $m_1+m_2+m_3=m-1$
  - $p^*(m_2+1)=t_y-t_x$,
  - $W'=W_{k-1}(t_u,t_x,m_1)+W_{k-1}(t_x,t_y,m_2)+W_{k-1}(t_y,t_v,m_3))+w_k$.

  Obviously, the subsets $U_{k-1}(t_u,t_x)$, $U_{k-1}(t_x,t_y)$ and $U_{k-1}(t_y,t_v)$ do not intersect. Thus, the JPS schedules of subsets that realize $W_{k-1}(t_u,t_x,m_1)$, $W_{k-1}(t_x,t_y,m_2)$ and $W_{k-1}(t_y,t_v,m_3)$, put one after another define a valid overall schedule of a

set of $m-1$ jobs taken in $U_{k-1}(t_u,t_v)$. Moreover, between t_x and t_y there is enough space to schedule J_k since m_2 jobs in $U_{k-1}(t_x,t_y)$ are scheduled and since $p*(m_2+1)=t_y-t_x$ (see Figure 2). As a consequence, we have $W' \leq W_k(t_u,t_v,m)$.

We now prove that $W_k(t_u,t_v,m) \leq W'$.

We only consider the case where $W_k(t_u,t_v,m)$ is finite otherwise the result holds. Consider a set Z that realizes $W_k(t_u,t_v,m)$. If J_k does not belong to Z then $W_k(t_u,t_v,m)=W_{k-1}(t_u,t_v,m) \leq W'$. Suppose now that $J_k \in Z$. Let t_x and t_y be the start and end times of J_k on the JPS of Z. Thanks to Proposition 3, we know that $t_x \in \Theta$ and $t_y \in \Theta$. We also have $\max(r_k,t_u) \leq t_x < t_y \leq \min(d_k,t_v)$. Let Z_1, Z_2, Z_3 be the partition of $Z-\{J_k\}$ into the jobs that have a release date between t_u and t_x, between t_x and t_y and between t_y and t_v. Because of the structure of JPS (J_k is the job whose due date is maximal), all jobs in Z_1 are completed before t_x. Moreover, all jobs in Z_2 start after t_x and are completed before t_y, and all jobs in Z_3 are completed before t_v. On top of that, $p*(|Z_2|+1)=t_y-t_x$ because J_k is also scheduled between t_x and t_y. Moreover, we have $|Z_1|+|Z_2|+|Z_3|+1=m$. Finally the weight of Z_1 is not greater than $W_{k-1}(t_u,t_x,|Z_1|)$, the weight of Z_2 is not greater than $W_{k-1}(t_x,t_y,|Z_2|)$ and the weight of Z_3 is not greater than $W_{k-1}(t_y,t_v,|Z_3|)$. This leads to $W_k(t_u,t_v,m) \leq W'$. □

Our dynamic programming algorithm relies on the above proposition. The values of $W_k(t_u,t_v,m)$ are stored in a multi-dimensional array of size $O(n^6)$ (n possible values for k, n^2 possible values for t_u, n^2 possible values for t_u, and n possible values for m).

- In the initialization phase the value of $W_1(t_u,t_v,m)$ is set to w_1 if $m = 1$ and if p is not greater than $\min(d_1,t_v)-\max(r_1,t_u)$ and to $-\infty$ otherwise.
- We then iterate from k=2 to k=n. Each time, W_k is computed for all the possible values of the parameters thanks to the formula of Proposition 4 and to the values of W_{k-1} computed at the previous step.

The maximum weighted number of on-time jobs is equal to:

$$\max(W_n(\min(t_i),\max(t_i),1),\ W_n(\min(t_i),\max(t_i),2),\ ...,\ W_n(\min(t_i),\max(t_i),n)).$$

The overall complexity of the algorithm is $O(n^5)$ for the initialization phase. For each value of k, $O(n^5)$ values of W_k have to be computed. For each of them, a maximum among $O(n^4)$ terms has to be computed (for given values of t_x, m_1 and m_2, there is only one possible value for both t_y and m_3). This leads to an overall time complexity of $O(n^{10})$. A rough analysis of the space complexity leads to an $O(n^6)$ bound but since, at each step of the outer loop on k, one only needs the values of W computed at the previous step (k-1), the algorithm can be implemented with 2 arrays of $O(n^5)$ size (one for the current values of W and one for the previous value of W).

4. Conclusion

In this paper, we have shown that two open scheduling problems are strongly polynomial. For both of them, we have provided some dynamic programming algorithms. Unfortunately, the worst case complexities of these algorithms are high and they do not appear to be suitable for real-life applications. However, there might be a scope for an improvement of these algorithms.

We believe that the study of the general non-preemptive problem $1|p_j,r_j|\Sigma w_j U_j$ is of great practical interest and we are currently trying to extend the branch and bound procedures initially designed for the non-weighted problem ([2], [15]) to the general weighted problem.

Acknowledgments

The author would like to thank Jacques Carlier for his insightful comments that allowed us to simplify several parts of this paper. We also thank Laurent Peridy for many enlightening discussions on the topics covered in this paper. The author is also grateful to Peter Brucker and Sigrid Knust whose work [4] has motivated this study.

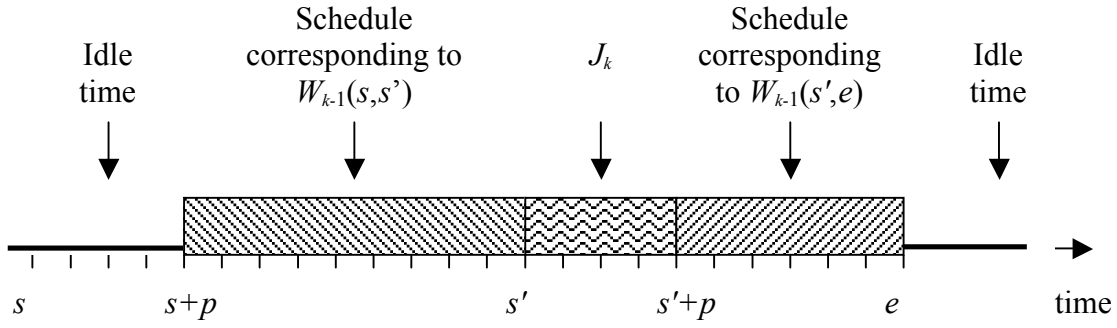

Figure 1. Illustration of Proposition 2

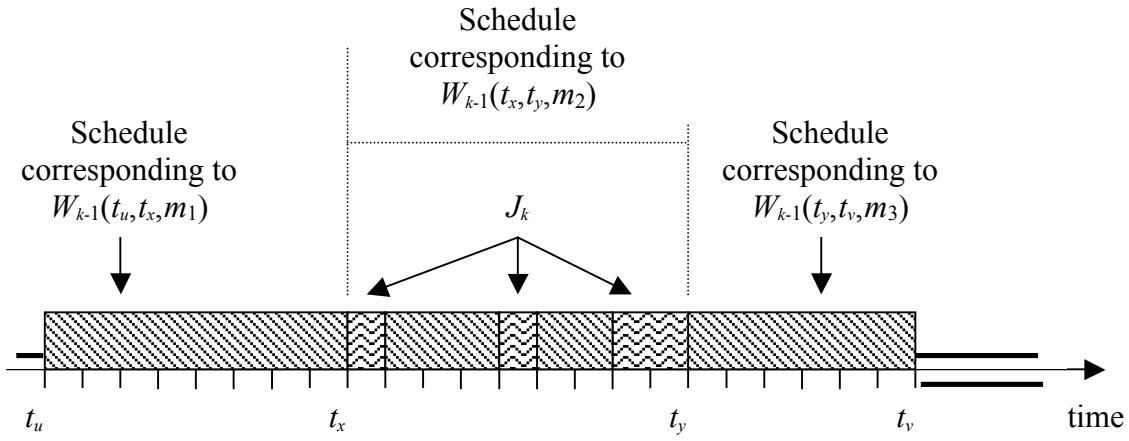

Figure 2. Illustration of Proposition 4

References

1. Philippe Baptiste, 'An $O(n^4)$ Algorithm for Preemptive Scheduling of a Single Machine to Minimize the Number of Late Jobs', *Technical Report* 98-98*, Université de Technologie de Compiègne* (1998); to appear in *Operations Research Letters*.

2. Philippe Baptiste, Claude Le Pape and Laurent Péridy, 'Global Constraints for Partial CSPs: A Case Study of Resource and Due-Date Constraints', to appear in the *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming, Pisa, Italy* (1998).

3. Peter Brucker, *Scheduling Algorithms*, Springer Lehrbuch, 1995.

4. Peter Brucker and Sigrid Knust, 'Complexity Results for Scheduling Problems', *URL: www//mathematik.uni-osnabrueck.de/research/OR/class*.

5. Jacques Carlier, *Problème à une machine et algorithmes polynômiaux.* QUESTIO, **5(4)**, 219-228 (1981).

6. Jacques Carlier, *Problèmes d'ordonnancements à contraintes de Ressources : Algorithmes et Complexité*, Thèse d'Etat, Paris VI, 1984.

7. Stéphane Dauzère-Pérès, 'Minimizing Late Jobs in the General One-Machine Scheduling Problem', *European Journal of Operational Research*, **81**, 134-142 (1995).

8. Stéphane Dauzère-Pérès and Marc Sevaux, 'Various Mathematical Programming Formulations for a General One Machine Sequencing Problem', *Rapport* 98/3/*AUTO*, *Ecole des Mines de Nantes* (1998).

9. Stéphane Dauzère-Pérès and Marc Sevaux, 'A Branch and Bound Method to Minimize the Number of Late Jobs on a Single Machine', *Rapport* 98/5/*AUTO*, *Ecole des Mines de Nantes* (1998).

10. Michael. R. Garey and David. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company (1979).

11. Hiroshi Kise, Toshihide Ibaraki and Hisashi Mine, 'A Solvable Case of the One-Machine Scheduling Problem with Ready and Due Times', *Operations Research* **26(1)**, 121-126 (1978).

12. Eugene L. Lawler, 'A Dynamic Programming Algorithm for Preemptive Scheduling of a Single Machine to Minimize the Number of Late Jobs', *Annals of Operations Research* **26**, 125-133 (1990).

13. Eugene L. Lawler, 'Knapsack-Like Scheduling Problems, the Moore-Hodgson Algorithm and the 'Tower of Sets' Property', *Mathl. Comput. Modelling*, **20(2)**, 91-106.

14. Michael J. Moore, "An *n* job, one machine sequencing algorithm for minimizing the number of late jobs", *Management Science* **15(1)**, 102-109 (1968).

15. Laurent Peridy, Philippe Baptiste and Eric Pinson, 'Branch and Bound Method for the Problem 1 | r_i | ΣU_i', *Proceedings of the* 6^{th} *international workshop on project management and scheduling* (1998).