

# Scheduling Equal-Length Jobs on Identical Parallel Machines

Philippe Baptiste<sup>1</sup>

HEUDIASYC, UMR CNRS 6599, Université de Technologie de Compiègne

Centre de Recherches de Royallieu

Rue Personne de Roberval, BP 20529, 60205 Compiègne Cedex, France

---

## Abstract

We study the situation where a set of  $n$  jobs with release dates and equal processing times have to be scheduled on  $m$  identical parallel machines. We show that, if the objective function can be expressed as the sum of  $n$  functions  $f_i$  of the completion time  $C_i$  of each job  $J_i$ , the problem can be solved in polynomial time for any fixed value of  $m$ . The only restriction is that functions  $f_i$  have to be non-decreasing and that for any pair of jobs  $(J_i, J_j)$ , the function  $f_i - f_j$  has to be monotonous. This assumption holds for several standard scheduling objectives, such as the weighted sum of completion times or the total tardiness. Hence, the problems  $(Pm|p_i = p, r_i|\sum w_i C_i)$  and  $(Pm|p_i = p, r_i|\sum T_i)$  are polynomially solvable.

*Key words:* Dynamic Programming; Parallel Machine Scheduling; Weighted Sum of Completion Times; Total Tardiness

---

---

<sup>1</sup> E-mail: Philippe.Baptiste@utc.fr.

## 1 Introduction

In this paper, we consider the scheduling situation where  $n$  jobs  $J_1, \dots, J_n$  have to be scheduled on  $m$  parallel identical machines. Each job is described by a release date  $r_i$ , a processing time  $p_i$  and a cost function  $f_i(t)$ . This function represents the cost induced by  $J_i$  when it is completed at time  $t$ . The problem of minimizing the sum of the  $f_i$  functions consists of finding a set of completion times  $C_i$  for each job  $J_i$  such that (1) jobs start after their release date, *i.e.*,  $\forall i, C_i - p_i \geq r_i$ , (2) no more than  $m$  machines are used at any time  $t$ , *i.e.*,  $\forall t, |\{J_i \mid C_i - p_i \leq t < C_i\}| \leq m$ , and (3) the objective function  $\sum_i f_i(C_i)$  is minimal. In the following, a schedule meeting (1) and (2) is said to be feasible.

This problem, denoted as the  $(P|r_i|\sum f_i(C_i))$  in the standard scheduling terminology (*e.g.*, [4]) is a generalization of several NP-hard scheduling problems.

As shown below, many scheduling criteria are formulated as a sum:

- Total weighted flow time.  $f_i(C_i) = w_i C_i$ ,  $w_i$  being the weight of the job  $J_i$  (*cf.*, left chart of Figure 1).
- Weighted number of late jobs.  $f_i(C_i) = w_i U_i$  where  $U_i$  equals 0 if  $C_i \leq d_i$  and equals 1 otherwise, the parameter  $d_i$  being the due date of the job  $J_i$  (*cf.*, middle chart of Figure 1).
- Total weighted tardiness.  $f_i(C_i) = w_i T_i$  where  $T_i$  equals  $\max(0, C_i - d_i)$  (*cf.*, right chart of Figure 1 for the non-weighted case.)

We refer to [3,4] for up to date complexity results on machine scheduling. In this paper, we study the special case where processing times are equal. As

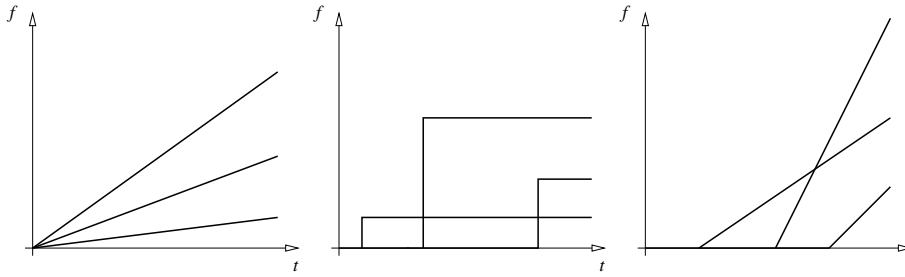


Fig. 1. Some cost functions  $f_i(t)$

shown in the literature, this assumption sometimes allows to exhibit polynomial time algorithms for problems that are NP-hard in the general case:

- Scheduling identical jobs on uniform parallel machines (*i.e.*, on machines that do not run at the same speed), is polynomial when release dates are equal and when the scheduling criteria is non-decreasing in the job completion times (see for instance [7]). In the case of distinct release dates, Dessouky, Lageweg, Lenstra and van de Velde show ([7]) that both the problem of minimizing makespan and the problem of minimizing the sum of the completion times (for a fixed number  $m$  of uniform machines) are polynomial.
- It is shown in [5,6,9] that the problem of minimizing the number of late jobs ( $1|r_i|\sum U_i$ ) is polynomial as soon as processing times are equal, while the general problem is NP-hard [8].
- Baptiste ([2]) shows that minimizing the weighted number of late jobs to be scheduled on a single machine can be done in polynomial time if processing times are equal. Two algorithms are presented, one for the non-preemptive problem ( $1|p_i = p, r_i|\sum w_i U_i$ ) and one for the preemptive one ( $1|pmtn, p_i = p, r_i|\sum w_i U_i$ ). Both problems being NP-Hard in the general case.

- Simons ([11]) provides a polynomial algorithm to minimize the sum of the completion times when processing times are equal ( $P|p_i = p, r_i| \sum C_i$ ) while the simple problem ( $1|r_i| \sum C_i$ ) is NP-Hard [10].

We show that for each fixed value of  $m$ , the problem with equal processing times is solvable in polynomial time provided that Hypothesis 1 holds:

**Hypothesis 1** *The functions  $f_i$  are non-decreasing, i.e.,  $\forall t_1, \forall t_2 > t_1, f_i(t_1) \leq f_i(t_2)$  and the functions  $f_i - f_j$  are monotonous, i.e., either  $\forall t_1, \forall t_2 > t_1, (f_i - f_j)(t_1) \leq (f_i - f_j)(t_2)$  or  $\forall t_1, \forall t_2 > t_1, (f_i - f_j)(t_1) \geq (f_i - f_j)(t_2)$ .*

It is easy to verify that Hypothesis 1 holds when minimizing, the weighted total completion time (left chart of Figure 1) or the total tardiness (special case of the right chart of Figure 1). Hence we show in this paper that the problems ( $Pm|p_i = p, r_i| \sum w_i C_i$ ), and ( $Pm|p_i = p, r_i| \sum T_i$ ), are polynomial, while their status was open, even for  $m = 1$ . From now on, we suppose that the above assumption hold.

In Section 2, we introduce a strict total order between jobs and we highlight some properties of a particular optimal schedule. To simplify the presentation, we first study the special case where  $m = 1$  (Section 3). The generalization of this algorithm for any fixed value of  $m$  is described in Section 4. The time and space complexities of the general algorithm are respectively  $O(n^{3m+4})$  and  $O(n^{2m+2})$ . Finally, we draw some conclusions in Section 5.

## 2 Some Properties of an Optimal Schedule

We define a binary relation  $\succ$  between jobs. Given any pair of jobs  $(J_i, J_j)$ ,  $J_i \succ J_j$  if and only if the function  $f_i - f_j$  is (1) either strictly increasing between some time points or (2) constant and  $i < j$ .

**Lemma 2** *The binary relation  $\succ$  is a strict total order.*

**PROOF.** Given the definition of the relation,  $\forall J_i, \forall J_j, J_i \succ J_j \Rightarrow J_i \neq J_j$ . Hence we only have to prove (1) that  $\succ$  is transitive and (2) that for any pair of distinct jobs  $(J_i, J_j)$ , either  $J_i \succ J_j$  or  $J_j \succ J_i$ .

- (1) Let  $J_i, J_j, J_k$  be any jobs such that  $J_i \succ J_j$  and  $J_j \succ J_k$ . Because  $J_i \succ J_j$  and  $J_j \succ J_k$ , both  $(f_i - f_j)$  and  $(f_j - f_k)$  are non-decreasing. Hence  $(f_i - f_j) + (f_j - f_k) = f_i - f_k$  is also non-decreasing. If there are some time points between which it is strictly increasing then  $J_i \succ J_k$ . If not, it is constant, then both  $f_i - f_j$  and  $f_j - f_k$  are constant. Consequently,  $i < j$  and  $j < k$ ; which leads to  $J_i \succ J_k$ .
- (2) Let  $J_i$  and  $J_j$  be two distinct jobs.  $f_i - f_j$  is monotonous. Thus it either equals a constant value (and hence  $J_i \succ J_j$  if  $i < j$  and  $J_j \succ J_i$  otherwise) or it is strictly increasing between two time points ( $J_i \succ J_j$ ) or it is strictly decreasing between two time points ( $J_j \succ J_i$ ).  $\square$

According to Lemma 2, we can suppose, without any loss of generality, jobs are sorted according to the strict total order  $\succ$ , *i.e.*,  $J_1 \succ J_2 \succ \dots \succ J_n$ . In the

following,  $\mathcal{S}$  denotes the schedule, among optimal ones, that lexicographically minimizes the vector made of the completion times  $(C_1, C_2, \dots, C_n)$ . Lemma 3 provides a characterization of the time points at which jobs start and end on the schedule  $\mathcal{S}$ .

**Lemma 3** *The time points at which jobs start and end on the schedule  $\mathcal{S}$  belong to  $T = \{t : \exists r_i, \exists l \in \{0, \dots, n\}, t = r_i + lp\}$ .*

**PROOF.** Consider the schedule  $\mathcal{S}$ . Let  $J_k$  be any job and let  $t$  be the largest time point, before the start time of  $J_k$ , at which the machine  $M$  on which  $J_k$  executes, is idle immediately before  $t$ . If  $t$  is not a release date, the job scheduled immediately after  $t$  on  $M$  can be scheduled earlier. Since the functions  $f_i$  are non-decreasing, the resulting schedule is still an optimal schedule. This contradicts the fact that  $\mathcal{S}$  is the optimal schedule on which the vector made of the completion times is lexicographically minimum.  $t$  is then a release date, say  $r_i$ . Between  $r_i$  and the starting time of  $J_k$ ,  $l$  jobs execute ( $0 \leq l \leq n - 1$ ). Hence the starting time and the ending time of  $J_k$  belong to  $T$ .  $\square$

### 3 The single machine case ( $m = 1$ )

We first introduce the variables of the dynamic programming algorithm and the Theorem that links them together (Section 3.1). The dynamic programming algorithm itself is described in Section 3.2.

### 3.1 Variables Definition and Decomposition Scheme

For any integer  $k \leq n$ , 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  $[s, e)$ . Let  $F_k(s, e)$  be the minimal value that the function  $\sum_{J_i \in U_k(s-p, e)} f_i(C_i)$  can take among the feasible schedules  $\mathcal{H}$  of all the jobs in  $U_k(s-p, e)$  such that

- (1) no machine is used before  $s$  on  $\mathcal{H}$ ,
- (2) no machine is used after  $e$  on  $\mathcal{H}$ ,
- (3) starting times of jobs on  $\mathcal{H}$  belong to  $T$ .

If no such schedule  $\mathcal{H}$  exists,  $F_k(s, e)$  is equal to  $\infty$ . Notice that given our definition,  $F_0(s, e)$  is equal to 0.

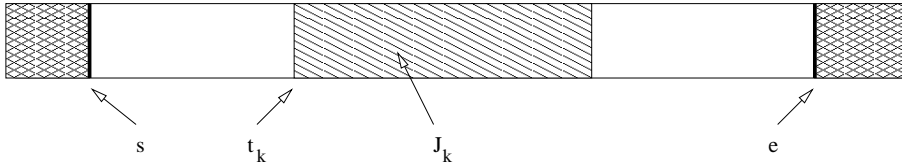


Fig. 2. Relative positions of  $J_k$  (starting at  $t_k$ ),  $s$  and  $e$

**Theorem 4** For any value of  $k$  in  $[1, n]$  and for any values  $s, e$  with  $s \leq e$ ,

$F_k(s, e)$  is equal to  $F_{k-1}(s, e)$  if  $r_k \notin [s-p, e)$  and otherwise to:

$$\left\{ \begin{array}{l} \min (F_{k-1}(s, t_k) + F_{k-1}(t_k + p, e) + f_k(t_k + p)) \\ t_k \in T \\ \max(s, r_k) \leq t_k \leq e - p \end{array} \right.$$

**PROOF.** Let  $F'$  be the expression above. If  $r_k \notin [s - p, e)$  the theorem holds because  $U_k(s - p, e) = U_{k-1}(s - p, e)$ . We assume that  $r_k \in [s - p, e)$ .

**We first prove that  $F' \geq F_k(s, e)$ .** We suppose that  $F'$  is finite otherwise the theorem holds. According to the definition of  $F'$ , there exists a time point  $t_k$  such that (1)  $t_k \geq \max(s, r_k)$ , (2)  $t_k \leq e - p$  and (3)  $F' = F_{k-1}(s, t_k) + F_{k-1}(t_k, e) + f_k(t_k + p)$ . Since  $F'$  is finite, there is a schedule  $\mathcal{H}_1$  that realizes  $F_{k-1}(s, t_k)$  and a schedule  $\mathcal{H}_2$  that realizes  $F_{k-1}(t_k + p, e)$ . Notice that any job in  $U_{k-1}(s - p, e)$  is either scheduled on  $\mathcal{H}_1$  or on  $\mathcal{H}_2$ . Consider the schedule  $\mathcal{H}$  build as follows: schedule  $J_k$  at time  $t_k$  and all other jobs of  $U_k(s - p, e)$  at the time they were scheduled on  $\mathcal{H}_1$  or on  $\mathcal{H}_2$ . Given the definition of  $F_{k-1}$ ,  $\mathcal{H}$  is a feasible schedule of  $U_k(s - p, e)$ , *i.e.*, jobs do not overlap in time and start (after their release dates) at time points that obviously belong to  $T$ . On top of that,  $\mathcal{H}$  is idle before  $s$  and after  $e$ . The value taken by the objective function on  $\mathcal{H}$  is exactly  $F'$  and hence,  $F' \geq F_k(s, e)$ .

**We now prove that  $F' \leq F_k(s, e)$ .** Suppose that  $F_k(s, e)$  is finite (otherwise the theorem holds). Among schedules that realize  $F_k(s, e)$ , let  $\mathcal{H}$  be the one on which the vector made of the completion times of the jobs in  $U_k(s - p, e)$ , taken in increasing order of index, is lexicographically minimum. Let  $t_k \in T$  be the starting time of  $J_k$  on  $\mathcal{H}$  ( $J_k$  is scheduled on  $\mathcal{H}$  because  $r_k \in [s - p, e)$ ). Suppose that there is a job  $J_i$  with  $r_i \leq t_k$  that is executed after  $J_k$ , at time  $t_i$  on  $\mathcal{H}$  ( $t_k \leq t_i$ ). Let  $\mathcal{H}'$  be the schedule obtained from  $\mathcal{H}$  by exchanging the jobs  $J_i$  and  $J_k$ . The variation  $\delta$  of the objective function induced by the

exchange is:

$$\begin{aligned}\delta &= f_i(t_k + p) + f_k(t_i + p) - (f_i(t_i + p) + f_k(t_k + p)) \\ &= (f_i - f_k)(t_k + p) - (f_i - f_k)(t_i + p)\end{aligned}$$

Since  $i < k$ ,  $J_i \succ J_k$  and consequently  $f_i - f_k$  is non-decreasing. Hence  $\delta \leq 0$ . Notice that given the definition of  $F_k(s, e)$ ,  $\delta$  cannot be strictly negative (this would contradict the fact that  $\mathcal{H}$  realizes the minimum). Both  $\mathcal{H}$  and  $\mathcal{H}'$  realize the same optimum value. However,  $\mathcal{H}'$  is better than  $\mathcal{H}$  for the lexicographical order. This contradicts our hypothesis on  $\mathcal{H}$ .

All jobs with a release date lower than or equal to  $t_k$  are scheduled before  $t_k$ . Stated another way, all jobs in  $U_{k-1}(s - p, t_k)$  are scheduled before  $t_k$  on  $\mathcal{H}$  and thus on  $\mathcal{H}$ , the sum over these jobs of the  $f_i$  functions is greater than  $F_{k-1}(s, t_k)$ . Similarly, all jobs in  $U_{k-1}(t_k, e)$  are scheduled after  $t_k + p$  and thus on  $\mathcal{H}$ , the sum over these jobs of the  $f_i$  functions is greater than  $F_{k-1}(t_k + p, e)$ . Let us compute  $F_k(s, e)$ . In the following  $C_i(\mathcal{H})$  denotes the completion time of the job  $J_i$  on the schedule  $\mathcal{H}$ .

$$\begin{aligned}F_k(s, e) &= \sum_{J_i \in U_k(s-p, e)} f_i(C_i(\mathcal{H})) \\ &= \sum_{J_i \in U_{k-1}(s-p, t_k)} f_i(C_i(\mathcal{H})) + \sum_{J_i \in U_{k-1}(t_k, e)} f_i(C_i(\mathcal{H})) + f_k(t_k + p) \\ &\geq F_{k-1}(s, t_k) + F_{k-1}(t_k + p, e) + f_k(t_k + p)\end{aligned}$$

Hence  $F_k(s, e) \geq F'$ .  $\square$

### 3.2 A Dynamic Programming Algorithm

On the optimal schedule  $\mathcal{S}$ , starting times belong to  $T$ . Consequently, the optimum is exactly  $F_n(\min_{t \in T} t, \max_{t \in T} t)$ . Thanks to Theorem 4, we have a straight dynamic programming algorithm to reach the optimum. The relevant values for  $s$  and  $e$  are exactly those in  $T$ . The values of  $F_k(s, e)$  are stored in a multi-dimensional array of size  $O(n^5)$  ( $n$  possible values for  $k$ ,  $n^2$  possible values both for  $s$  and  $e$ ). Our algorithm works as follows:

- In the initialization phase,  $F_0(s, e)$  is set to 0 for any values  $s, e$  in  $T$  ( $s \leq e$ ).
- We then iterate from  $k = 1$  to  $k = n$ . Each time,  $F_k$  is computed for all the possible values of the parameters thanks to the formula of Theorem 4, and to the values of  $F_{k-1}$  computed at the previous step.

The initialization phase runs in  $O(n^4)$  because the size of  $T$  is upper bounded by  $n^2$ . Afterwards, for each value of  $k$ ,  $O(n^4)$  values of  $F_k(s, e)$  have to be computed. For each of them, a maximum among  $O(n^2)$  terms is computed (because there are  $O(n^2)$  possible values for  $t_k \in T$ ). 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  $F$  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  $F$  and one for the previous values of  $F$ . (To build the optimal schedule, all values of  $F_k(s, e)$  have to be kept; hence the initial  $O(n^5)$  bound applies.)

## 4 The Cumulative Case ( $m > 1$ )

The structure of this section is the same as for the single machine case. The variables of the dynamic programming algorithm and the Theorem that links them together are introduced in Section 4.1. The dynamic programming algorithm itself is described in Section 4.2.

### 4.1 Variables Definition and Decomposition Scheme

Roughly speaking, the decomposition scheme for the single machine case (Theorem 4) relies on a particular time point  $t_k \in [s, e)$  that allows us to split the problem into two sub-problems (between  $s$  and  $t_k$  and between  $t_k$  and  $e$ ). The same scheme is kept here, however  $s$  and  $e$  are replaced by two vectors  $\sigma$  and  $\epsilon$  that represent some resource profiles. We introduce more formally this notion.

**Definition 5** *A resource profile  $\xi$  is a vector  $(\xi_1, \xi_2, \dots, \xi_m)$  such that  $\xi_1 \leq \xi_2 \leq \dots \leq \xi_m$  and  $\xi_m - \xi_1 \leq p$ . In the following,  $\Xi$  denotes the set of resource profiles  $\xi$  such that  $\forall i, \xi_i \in T$ .*

**Intuitive meaning of resource profiles.** A resource profile  $\xi = (\xi_1, \xi_2, \dots, \xi_m)$  represents the state of the resource at some time point  $\xi_1$  (in practice this time point always matches the starting time or the completion time of a job). If  $q$  machines are idle at time  $\xi_1$  then the  $q$  first components of the resource profile equal  $\xi_1$ . The other components are the completion times of the  $m - q$  jobs that are being processed at time  $\xi_1$ . Since the processing time of jobs is  $p$ ,

we have  $\forall i, \xi_i - \xi_1 \leq p$ . The state of the resource is considered from a global point of view and thus, the  $i^{\text{th}}$  component of the resource profile is not systematically related to the  $i^{\text{th}}$  machine. In the following, some “left” resource profiles  $\sigma$  are used to state that no machine is available before  $\sigma_1$ , one machine is available between  $\sigma_1$  and  $\sigma_2$ ; two machines are available between  $\sigma_2$  and  $\sigma_3$ , *etc.* Conversely, some “right” resource profiles  $\epsilon$  are used to state that no machine is available after  $\epsilon_m$ , one machine is available between  $\epsilon_{m-1}$  and  $\epsilon_m$ ; two machines are available between  $\epsilon_{m-2}$  and  $\epsilon_{m-1}$ , *etc.* The resource profiles  $\sigma$  and  $\epsilon$  allow us to determine the exact amount of resource that is available at each time point in  $[\sigma_1, \epsilon_m]$ .

**Definition 6** *Given two resource profiles  $\xi$  and  $\mu$ ,  $\xi \ll \mu$  if and only if for any index  $i$  in  $\{1, \dots, m\}$ ,  $\xi_i \leq \mu_i$ .*

Notice that  $\ll$  defines a partial order on resource profiles. We introduce a technical lemma that will be used later on.

**Lemma 7** *Given two resource profiles  $\xi$  and  $\mu$ ,  $\xi \ll \mu$  if and only if for any time point  $t$ ,  $|\{i : t < \xi_i\}| + |\{i : \mu_i \leq t\}| \leq m$ .*

**PROOF.**

**Sufficient condition.** (By contradiction.) Let  $k$  be the first index such that  $\mu_k < \xi_k$ . Let us compute  $|\{i : t < \xi_i\}| + |\{i : \mu_i \leq t\}|$  for  $t = \mu_k$ . It is equal to  $|\{i : \mu_k < \xi_i\}|$  plus  $|\{i : \mu_i \leq \mu_k\}|$ ; which is greater than or equal to  $(m - k + 1) + k > m$ . This contradicts our hypothesis.

**Necessary condition.** Let  $t$  be any time point. In the following,  $\delta(P)$  denotes

the binary variable that equals 1 if the condition  $P$  holds, 0 otherwise. Notice that for any value of  $i$ ,  $\delta(t < \xi_i) + \delta(\mu_i \leq t) \leq 1$  because  $\xi_i \leq \mu_i$ . Hence,

$$|\{i : t < \xi_i\}| + |\{i : \mu_i \leq t\}| = \sum_{i=1}^m (\delta(t < \xi_i) + \delta(\mu_i \leq t)) \leq m. \quad \square$$

We now define the variables of the dynamic programming algorithm. For any integer  $k \leq n$ , for any resource profiles  $\sigma$  and  $\epsilon$  ( $\sigma \ll \epsilon$ ), let  $F_k(\sigma, \epsilon)$  be the minimal value that the function  $\sum_{J_i \in U_k(\sigma_m - p, \epsilon_1)} f_i(C_i)$  can take among the schedules of all the jobs in  $U_k(\sigma_m - p, \epsilon_1)$  such that

- starting times belong to  $T$ ,
- the number of machines available at time  $t$  to schedule the jobs in the set  $U_k(\sigma_m - p, \epsilon_1)$  is  $m - |\{i : t < \sigma_i\}| - |\{i : \epsilon_i \leq t\}|$ .

If no such schedule exists,  $F_k(\sigma, \epsilon)$  is equal to  $\infty$ . Notice that given our definition,  $F_0(\sigma, \epsilon)$  is equal to 0.

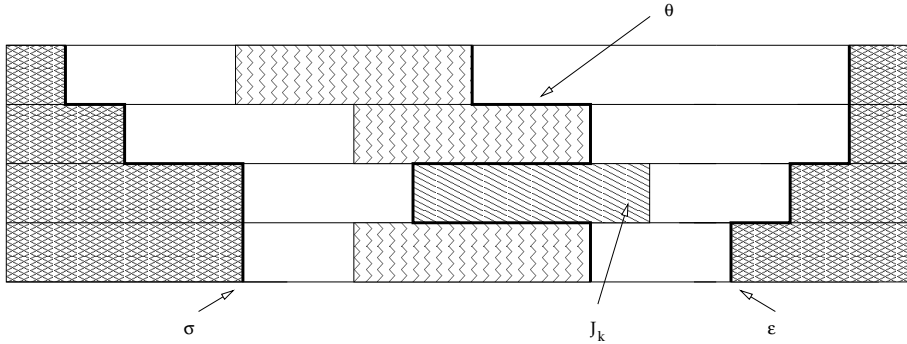


Fig. 3. Resource profiles

**Theorem 8** For any value of  $k$  in  $[1, n]$ , for any resource profiles  $\sigma$  and  $\epsilon$  such

that  $\sigma \ll \epsilon$ ,  $F_k(\sigma, \epsilon)$  is equal to  $F_{k-1}(\sigma, \epsilon)$  if  $r_k \notin [\sigma_m - p, \epsilon_1)$  and otherwise to:

$$\min \left( F_{k-1}(\sigma, \theta) + F_{k-1}(\theta', \epsilon) + f_k(\theta_1 + p) \right)$$

$$\left\{ \begin{array}{l} \theta \in \Xi, r_k \leq \theta_1, \sigma \ll \theta \\ \theta' = (\theta_2, \dots, \theta_m, \theta_1 + p) \ll \epsilon \end{array} \right.$$

Notice that in the above formula the value  $\theta'$  is derived from  $\theta$ . Figure 3 provides an illustration of this theorem. Theorem 8 basically states that the optimum schedule for  $k, \sigma, \epsilon$  can be computed by trying all possible resource profiles  $\theta$  of  $\Xi$  that are “between” the resource profiles  $\sigma$  and  $\epsilon$ . For each candidate resource profile  $\theta$ , the job  $J_k$  starts at  $\theta_1$ .

**PROOF.** Let  $F'$  be the expression above. First notice that,  $\theta'$  is a resource profile because  $\theta \in \Xi$ . Hence the use of  $F_{k-1}(\theta', \epsilon)$  is correct. If  $r_k \notin [\sigma_m - p, \epsilon_1)$  the proposition obviously holds because  $U_k(\sigma_m - p, \epsilon_1) = U_{k-1}(\sigma_m - p, \epsilon_1)$ . We now consider the case where  $r_k \in [\sigma_m - p, \epsilon_1)$ .

**We first prove that  $F' \geq F_k(\sigma, \epsilon)$ .** We suppose that  $F'$  is finite otherwise the proposition holds. Let  $\theta \in \Xi$  be the resource profile that realizes  $F'$ . There is a schedule  $\mathcal{H}_1$  that realizes  $F_{k-1}(\sigma, \theta)$  and a schedule  $\mathcal{H}_2$  that realizes  $F_{k-1}(\theta', \epsilon)$ . Notice that any job in  $U_{k-1}(\sigma_m - p, \epsilon_1)$  is either scheduled on  $\mathcal{H}_1$  or on  $\mathcal{H}_2$ . Consider the schedule  $\mathcal{H}$  build as follows: schedule  $J_k$  at time  $\theta_1$  and all other jobs in  $U_k(\sigma_m - p, \epsilon_1)$  at the time they were scheduled on  $\mathcal{H}_1$  or on  $\mathcal{H}_2$ . Let us prove that  $\mathcal{H}$  is a feasible schedule of  $U_k(\sigma_m - p, \epsilon_1)$ . Since  $r_k \leq \theta_1$  and since  $\mathcal{H}_1$  and  $\mathcal{H}_2$  are feasible, all jobs are scheduled after their release date.

Moreover, we claim that the resource constraint is satisfied.

Let  $t$  be any time point. The number of machines used by the jobs scheduled on  $\mathcal{H}_1$  is upper bounded by  $m - |\{i : t < \sigma_i\}| - |\{i : \theta_i \leq t\}|$ . The number of machines used by the jobs scheduled on  $\mathcal{H}_2$  is upper bounded by  $m - |\{i : t < \theta'_i\}| - |\{i : \epsilon_i \leq t\}|$ . Finally,  $J_k$  uses a machine at time  $t$  if and only if  $t \in [\theta_1, \theta_1 + p)$ . The resource constraint is satisfied at time  $t$  if the sum of the upper bounds is lower than or equal to  $m - |\{i : t < \sigma_i\}| - |\{i : \epsilon_i \leq t\}|$ , *i.e.*, if the following expression is lower than or equal to 0.

$$\begin{aligned}
& m - |\{i : t < \theta'_i\}| - |\{i : \theta_i \leq t\}| + \delta(\theta_1 \leq t < \theta_1 + p) \\
&= m - \sum_{i=1}^m \delta(t < \theta'_i) - \sum_{i=1}^m \delta(\theta_i \leq t) + \delta(\theta_1 \leq t < \theta_1 + p) \\
&= m - \left( \sum_{i=2}^m \delta(t < \theta_i) + \delta(t < \theta_1 + p) \right) - \sum_{i=1}^m \delta(\theta_i \leq t) + \delta(\theta_1 \leq t < \theta_1 + p) \\
&= m - \sum_{i=2}^m (\delta(t < \theta_i) + \delta(\theta_i \leq t)) - \delta(t < \theta_1 + p) - \delta(\theta_1 \leq t) + \delta(\theta_1 \leq t < \theta_1 + p) \\
&= 1 - \delta(t < \theta_1 + p) - \delta(\theta_1 \leq t) + \delta(\theta_1 \leq t < \theta_1 + p) \leq 0
\end{aligned}$$

We have proven that  $\mathcal{H}$  is a feasible schedule of  $U_{k-1}(\sigma_m - p, \epsilon_1)$ . On top of that, starting times obviously belong to  $T$ . The value taken by the objective function on  $\mathcal{H}$  is exactly  $F'$  and hence,  $F' \geq F_k(\sigma, \epsilon)$ .

**We now prove that  $F' \leq F_k(\sigma, \epsilon)$ .** Suppose that  $F_k(\sigma, \epsilon)$  is finite (otherwise the proposition holds). Among the schedules that realize  $F_k(\sigma, \epsilon)$ , let  $\mathcal{H}$  be the one on which the vector made of the completion times of the jobs in  $U_k(\sigma_m - p, \epsilon_1)$ , taken in increasing order of index, is lexicographically minimum. Let  $t_k$  be the starting time of  $J_k$  on  $\mathcal{H}$  ( $J_k$  is on  $\mathcal{H}$  because  $r_k \in [\sigma_m - p, \epsilon_1)$ ).

The proof works as follows. We first show that on  $\mathcal{H}$ , all jobs with a release date lower than or equal to  $t_k$  start before or at  $t_k$ . We then exhibit a resource profile  $\theta \in \Xi$  such that (1)  $\theta_1 = t_k$ , (2)  $\sigma \ll \theta$ , (3)  $\theta' = (\theta_2, \dots, \theta_m, \theta_1 + p) \ll \epsilon$ . We then conclude the proof.

On  $\mathcal{H}$ , jobs with a release date lower than or equal to  $t_k$  start before or at  $t_k$ .

Suppose that there is a job  $J_i$  with  $r_i \leq t_k$  that is executed at time  $t_i > t_k$  on  $\mathcal{H}$ . Let  $\mathcal{H}'$  be the schedule obtained from  $\mathcal{H}$  by exchanging the jobs  $J_i$  and  $J_k$ .  $\mathcal{H}'$  is better than  $\mathcal{H}$  (*cf.* proof of the second part of Theorem 4). This contradicts our hypothesis on  $\mathcal{H}$ .

Definition of  $\theta$ . Let  $\tau$  the vector build component per component as follows: The first component of  $\tau$  is  $t_k$ , the time at which  $J_k$  starts. The following components of  $\tau$  are the end times on  $\mathcal{H}$  of the jobs (except  $J_k$ ) that start before or at  $t_k$  and end strictly after  $t_k$ . The following components are the values  $\sigma_i$  that are strictly greater than  $t_k$ . Since the resource constraint holds at time  $t$  for  $\mathcal{H}$ , it is easy to prove that the dimension of  $\tau$  is lower than or equal to  $m$ . The vector  $\tau$  is extended to a vector  $\tau'$  of dimension  $m$  by adding a sufficient number of times a component  $t_k$ . Let  $\theta$  be the vector obtained from  $\tau'$  by sorting its components in increasing order.

$\theta$  belongs to  $\Xi$ . Consider a component  $\theta_j$  of  $\theta$ . Either it is the end time of a job and then  $t_k < \theta_j \leq t_k + p$ ) or it is a  $\sigma_i > t_k$  value (and then  $t_k < \theta_j \leq t_k + p$  otherwise  $t_k$  would be strictly lower than  $\sigma_1$  and hence no machine would be available at time  $t_k$ ) or it is equal to  $t_k$ . Hence, all components belong to the interval  $[t_k, t_k + p]$ , as a consequence  $\theta_m - \theta_1 \leq p$ . We have proven that  $\theta$  is a resource profile. It is also easy to verify that all components of  $\theta$  belong to

$T$  and hence  $\theta \in \Xi$ . On top of that, it is obvious that  $\theta_1 = t_k$ . The proof that  $\sigma \ll \theta$  is also immediate given the definition of  $\theta$ .

$\theta' = (\theta_2, \dots, \theta_m, \theta_1 + p) \ll \epsilon$ . The fact that  $\theta'$  is a resource profile comes immediately from  $\theta \in \Xi$ . Suppose that the relation  $\theta' \ll \epsilon$  does not hold. Then, according to Lemma 7, there is a time point  $t$  such that  $|\{i : t < \theta'_i\}| + |\{i : \epsilon_i \leq t\}| > m$ . Recall that  $\epsilon_1 \geq t_k$  otherwise no machine would be available at the time point where  $J_k$  ends. Hence, if  $t < t_k$ ,  $|\{i : \epsilon_i \leq t\}| = 0$  and consequently,  $|\{i : t < \theta'_i\}| > m$ ; which contradicts the fact that  $\theta'$  is a vector of dimension  $m$ . As a consequence, we have  $t_k \leq t$ . Let  $O$  be the set of jobs that start before or at  $t_k$  and end strictly after  $t_k$ . The components of  $\theta'$ , are either the completion times of the jobs in  $O$  or the  $\sigma_i$  values that are strictly greater than  $t_k$  or are equal to  $t_k$ . Hence, the number of components of  $\theta'$  that are strictly greater than  $t$  is equal to the sum of (1) the number of jobs in  $O$  that end strictly after time  $t$  and of (2) the number of components of  $\sigma_i$  that are strictly greater than  $t$ . Since  $t_k \leq t$ , the jobs in  $O$  all start before  $t$ . Hence, the total number of jobs  $N_t$  that start before or at  $t$  and end strictly after  $t$  plus the number of components of  $\sigma_i$  that are strictly greater than  $t$ , is greater than or equal to  $|\{i : t < \theta'_i\}|$ . Hence,  $|\{i : t < \theta'_i\}| + |\{i : \epsilon_i \leq t\}| > m$  leads to,  $N_t + |\{i : t < \sigma_i\}| + |\{i : \epsilon_i \leq t\}| > m$ . This contradicts the fact that the resource constraint is met at time  $t$  on  $\mathcal{H}$ .

Conclusion.  $\mathcal{H}$ , restricted to the jobs with a release date lower than or equal to  $t_k$  (except  $J_k$ ), realizes  $F_{k-1}(\sigma, \theta)$  (otherwise,  $F_k(\sigma, \epsilon)$  would not be optimum). Similarly,  $\mathcal{H}$  restricted to the jobs with a release date strictly greater than  $t_k$  realizes  $F_{k-1}(\theta', \epsilon)$ . Hence,  $F_k(\sigma, \epsilon) = F_{k-1}(\sigma, \theta) + F_{k-1}(\theta', \epsilon) + f_k(\theta_1 + p)$ .  $\square$

#### 4.2 A Dynamic Programming Algorithm

Given the fact that on  $\mathcal{S}$ , starting times belong to  $T$ , the optimum is exactly  $F_n((\min_{t \in T} t, \dots, \min_{t \in T} t), (\max_{t \in T} t, \dots, \max_{t \in T} t))$ . Thanks to Theorem 8, we have a straight dynamic programming algorithm to compute this value. The relevant values for  $\sigma$  and  $\epsilon$  are exactly the vectors in  $\Xi$ . We claim that there are  $O(n^2 n^{m-1}) = O(n^{m+1})$  relevant resource profiles. Indeed, there are  $n^2$  possible values for the first component and once it is fixed there are only  $n$  possible choices for the  $m - 1$  remaining ones (because of the structure of  $T$  and because the difference between the  $m$ -th component and the first one is upper bounded by  $p$ ). This means that there are  $O(n^{2m+2})$  relevant pairs  $(\sigma, \epsilon)$ . The values of  $F_k(\sigma, \epsilon)$  are stored in a multi-dimensional array of size  $O(n^{2m+3})$  ( $n$  possible values for  $k$ ,  $n^{m+1}$  possible values for  $\sigma$  and  $n^{m+1}$  possible values for  $\epsilon$ ). Our algorithm then works as follows.

- In the initialization phase,  $F_0(\sigma, \epsilon)$  is set to 0 for any values  $\sigma, \epsilon$  in  $\Xi$  such that  $\sigma \ll \epsilon$ .
- We then iterate from  $k = 1$  to  $k = n$ . Each time,  $F_k$  is computed for all the possible values of the parameters thanks to the formula of Proposition 8, and to the values of  $F_{k-1}$  computed at the previous step.

Before analyzing the complexity of the overall algorithm, remark that one can generate easily all possible resource profiles  $\theta$  between  $\sigma$  and  $\epsilon$  (*i.e.*,  $\sigma \ll \theta \ll \epsilon$ ) in  $O(n^{m+1})$  steps. Indeed, there are  $O(n^2)$  possible values  $\theta_1 \in T \cap [\sigma_1, \epsilon_1]$ . The other components of  $\theta$  belong to  $T \cap [\theta_1, \theta_1 + p]$ . There are only  $O(n)$  values

in this set. Components  $\theta_i$  are generated, one after another; each time a test verifying that  $\sigma_i \leq \theta_i \leq \epsilon_i$  being performed in constant time.

In the initialization phase,  $O(n^{2m+2})$  pairs ( $\sigma \ll \epsilon$ ) are generated. Afterwards, for each value of  $k$ ,  $O(n^{2m+2})$  values of  $F_k$  have to be computed. For each of them,  $O(n^{m+1})$  resource profiles  $\theta$  are generated with  $\sigma \ll \theta \ll \epsilon$ . A minimum among  $O(n^{m+1})$  terms is computed. This leads to an overall time complexity of  $O(n^{3m+4})$ . A rough analysis of the space complexity leads to an  $O(n^{2m+3})$  bound but since, at each step of the outer loop on  $k$ , one only needs the values of  $F$  computed at the previous step ( $k-1$ ), the algorithm can be implemented with 2 arrays of  $O(n^{2m+2})$  size: one for the current values of  $F$  and one for the previous values of  $F$ .

Notice that we can perform a backward computation on the values  $F_k(\sigma, \epsilon)$  to recover the optimum schedule. Indeed, for all relevant values of  $k, \sigma$  and  $\epsilon$ , it is easy to identify which resource profile  $\theta$  is the best (according to the fundamental recursion formula of Theorem 8). Since the starting time of  $J_k$  is the first component  $\theta_1$  of  $\theta$ , we can recover the starting times of all jobs at once.

## 5 Conclusion

We have shown that scheduling equal length jobs on a fixed number of parallel identical machines is a polynomial problem for some objective functions. We have established that several open scheduling problems such as  $(Pm|p_j = p, r_j|\sum_j w_j C_j)$  or  $(Pm|p_j = p, r_j|\sum_j T_j)$  are polynomial. At this point, we

think that the algorithms described in this paper can be somewhat generalized.

- The structure of the algorithm for the single machine case, is close to the one used in [2] for minimizing the weighted number of late jobs when processing times are equal ( $1|p_j = p, r_j|\sum_j w_j U_j$ ). However, the functions  $w_j U_j$  do not verify Hypothesis 1. It makes us think that there might exist a less restrictive class of functions  $f_i$  for which the problem remains polynomial.
- Most of the results described in this paper can be extended to the case of uniform machines, *i.e.*, machines that do not run at the same speed. This remains however to be formally proven.

An interesting open question concerns the status of the problems when the number of machines is not fixed. Finally, we would like to mention that the status of the preemptive versions of the problems studied in this paper are open. A polynomial time algorithm has been proposed in [2] for the ( $1|p_j = p, r_j, pmtn|\sum_j w_j U_j$ ). Despite our efforts, we have not been able to generalize this algorithm to a larger class of scheduling problems.

## Acknowledgements

The author would like to thank Jacques Carlier and Peter Brucker for their helpful comments. Special thanks to Steef van de Velde and Maged Dessouky for the papers they made available for us.

## References

- [1] Kenneth R. Baker, Introduction to sequencing and scheduling, Wiley (New-York, 1974).
- [2] Philippe Baptiste, Polynomial Time Algorithms for Minimizing the Weighted Number of Late Jobs on a Single Machine when Processing Times are Equal, Technical Report UTC 138(1998), to appear in Journal of Scheduling.
- [3] Peter Brucker and Sigrid Knust, Complexity Results of Scheduling Problems, URL: [www//mathematik.uni-osnabrueck.de/research/OR/class](http://mathematik.uni-osnabrueck.de/research/OR/class).
- [4] Peter Brucker, Scheduling Algorithms Springer Lehrbuch (1995).
- [5] Jacques Carlier, Problème à une machine et algorithmes polynômiaux. *QUESTIO* 5 (1981) 219-228.
- [6] Jacques Carlier, Problèmes d'ordonnancements contraintes de Ressources : Algorithmes et Complexité, Thèse d'Etat, Université Paris VI (1984).
- [7] Mohamed I. Dessouky, Ben J. Lageweg, Jan Karel Lenstra and Steef L. van de Velde, Scheduling identical jobs on uniform parallel machines, *Statistica Neerlandica* 44 (1990) 115-123.
- [8] Michael. R. Garey and David. S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, W. H. Freeman and Company (1979).
- [9] Michael. R. Garey, David. S. Johnson, Barbara B. Simons and R.E. Tarjan, Scheduling unit-time tasks with arbitrary release times and deadlines, *SIAM Journal of Computing*, 10 (1981) 256-269.

- [10] Jan Karel Lenstra, Alexander H.G. Rinnooy Kan and Peter Brucker, Complexity of machine scheduling problems, *Annals of Discrete Mathematics*, 1 (1977) 343-362.
- [11] Barbara Simons, Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines, *SIAM Journal of Computing* 12 (1983) 294-299.