Single-machine scheduling with time window-dependent processing times

Chams Lahlou, Stéphane Dauzère-Pérès

Single machine scheduling with time window dependent processing times

Chams Lahlou Stéphane Dauzère-Pérès

IRCCyN - Ecole des Mines de Nantes
CNRS, UMR 6597
La Chantrerie – BP 20722
F-44307 Nantes Cedex 3 – France
Email: Chams.Lahlou@emn.fr, Stephane.Dauzere-Peres@emn.fr

ABSTRACT

In the one-machine scheduling problems analysed in this paper, the processing time of a job depends on the time at which the job is started. More precisely, the horizon is divided into time windows and with each one a coefficient is associated that is used to determine the actual processing time of a job starting in it. Two models are introduced, and one of them has direct connections with models considered in previous papers on scheduling problems with time dependent processing times. Various computational complexity results are presented for the makespan criterion, which show that the problem is NP-hard, even with 2 time windows. Solving procedures are also proposed for some special cases.

Keywords: scheduling; one-machine; time dependent processing time; time windows; computational complexity.

Introduction

One of the most classic problems in scheduling theory consists in scheduling a set $\mathcal{J} = \{1, \ldots, n\}$ of independent jobs on a single machine. Preemption is not allowed (a job cannot be interrupted while being processed) and no more than one job can be processed at a time. Processing times of the jobs are usually considered to be given and constant. However,

in practical settings, the processing time of a job may depend on the time at which an operation starts on a resource. Looking at the vast body of literature in scheduling, only a few papers have considered time dependent processing times. The first papers on the subject are cited in Gawiejnowicz [1]. A very extensive survey can be found in Alidaee and Womer [2], and a more recent survey is provided in Cheng *et al.* [3].

From now on, we only consider problems with the makespan criterion, i.e for the minimisation of the schedule length. When processing times are linear, $p_j = x_j + \alpha_j t_j$, where α_j is the rate at which the processing time of the job increases (or decreases, if $\alpha_j < 0$). The problem can be solved in $\mathcal{O}(n \log n)$ time. This result was proved independently by Tanaev *et al.* [4], Gupta and Gupta [5], Browne and Yechali [6] and Gawiejnowicz and Pandowska [7].

In the case of piecewise linear processing times, three models have been studied. In the model proposed by Sundararaghavan and Kunnathur [8], $p_j = x_j$ if $t_j \leq d$, where d is a given date, and $p_j = x_j + \alpha_j$ else. The problem has been shown to be binary NP-hard by Mosheiov [9], whereas Alidaee and Womer [2] have shown that it can be solved in time $\mathcal{O}(n \log n)$ if the x_j's are identical. In a more general model introduced by Mosheiov [9], the processing times are subject to multi-step deteriorations, i.e each job j has several dates $d_{1,j} < d_{2,j} < \cdots < d_{w,j}$ and coefficients $\alpha_{1,j} < \alpha_{2,j} < \cdots < \alpha_{w,j}$ such that $p_j = x_j + \alpha_{i,j}$ if $d_{i-1,j} < t_j \leq d_{i,j}$. In the third model, there are two given dates, d and D, and $p_j = x_j$ if $t_j \leq d$, $p_j = x_j + \alpha_j(t_j - d)$ if $d < t_j < D$, and $p_j = \alpha_j(D - d)$ otherwise. The problem has been shown to be binary NP-hard by Kononov [10], Cai *et al.* [11] and Kubiak and van de Velde [12], even if $D = \infty$. For the case where $D = \infty$, a pseudopolynomial time algorithm that runs in $\mathcal{O}(nd \sum x_j)$ and a branch-and-bound algorithm have been proposed by Kubiak and van de Velde [12]. In the same paper, two pseudopolynomial time algorithms are proposed for the case where $D < \infty$.

Since the problems studied in this paper have few connections with the nonlinear pro-

cessing time case, the reader is referred to the paper of Alidae and Womer [2].

In our problem, a sequence of dates $(d_1, \ldots, d_{w+1})$ defines w time windows $[d_i, d_{i+1}[$ where $d_i < d_{i+1}$, $d_1 = 0$ and $d_{w+1} = +\infty$. With each job j is associated a normal processing time x_j and w coefficients α_{ij}, one for each time window. The actual processing time p_j of a job j depends on the coefficient of the time window in which it starts. For a job j starting at time t_j, we define the two following models:

- Model M^+: $p_j = x_j + \alpha_{ij}$ if $d_i \leq t_j < d_{i+1}$.
- Model $M^\times$: $p_j = \alpha_{ij} x_j$ if $d_i \leq t_j < d_{i+1}$,

The objective is to find a schedule that minimises the makespan $C_{\max} = \max_{j \in \mathcal{J}} C_j$, where $C_j = t_j + p_j$ is the completion time of job j.

In model M^+, the coefficient α_{ij} models a waste of or a saving on the normal processing time of job j when it starts in time window $[d_i, d_{i+1}[$: job j can take more time (i.e $\alpha_{ij} > 0$), less time (i.e $\alpha_{ij} < 0$) or the same time (i.e $\alpha_{ij} = 0$) than the normal processing time. This model has some connections with the two first piecewise linear processing time models we have presented. First, it is a generalisation of the one proposed by Sundaraghavan and Kunnathur [8], where $p_j = x_j$ if $t_j \leq d$ and $p_j = x_j + \alpha_j$ otherwise. Indeed, to get their model we just have to consider two time windows such that $\alpha_{1j} = 0$, $\alpha_{2j} = \alpha_j$ and $d_2 = d + \epsilon$ (with ϵ being strictly positive and smaller than the smallest processing time in a window). First, the case of M^+ where $\alpha_i < \alpha_{i+1}$ is equivalent to a special case of the model proposed by Mosheiov [9] where $d_{i,j} = d_i$, for all j. There are actually two differences between Mosheiov's model and M^+: (i) contrary to Mosheiov's model where only deteriorating processing times are considered, any kind of step function can be used in M^+ to model the processing times; (ii) in Mosheiov's model, each job has its own sequence of time windows whereas in our model the jobs share the same sequence of time windows.

The model M^+ can be useful to solve practical scheduling problems when the processing times depend on time periods. For instance, consider a workshop where the processing time of job j is x_j when all the workers are present (e.g. there are k workers), and suppose there is a delay p_j^- (a saving p_j^+) to process job j if the number of workers is smaller (larger) than k. The problem of scheduling the jobs in minimum time is modelled by associating with each day a time window, and with each pair (day i, job j) a coefficient α_{ij} according to the number of workers present in day i : we set $\alpha_{ij} = p_j^-$ if the number of workers is smaller than k, $\alpha_{ij} = -p_j^+$ if the number of workers is larger than k, and $\alpha_{ij} = 0$ otherwise.

Contrary to M^+, the model $M^\times$ has no direct link to previous models. However, it deserves to be studied since in some practical cases the processing time of a job is proportional to the availability of a resource. For instance, if in a network of processors the speed of a task depends linearly on the number of available processors, we set $p_j = x_j/n_i$, that is $\alpha_{ij} = \frac{1}{n_i}$, where n_i is the number of available processors in time period i. Moreover, as we shall see, there exist interesting differences between $M+$ and $M*$ from a complexity point of view.

Finally, another application of M^+ and $M^\times$ could be the approximation of non linear processing times (see the survey of Alidaee and Womer [2] for examples of this type of processing time), by using time windows as intervals for the discretisation of the non linear function.

In the next section, we present an optimal algorithm for both models when the sequence of jobs is given. Then, the $M^\times$ model with two time windows is studied: It is shown that the problem is NP-hard in the general case and can be solved using a pseudo-polynomial time algorithm when the coefficient of a time window is the same for each job, i.e $\alpha_{ij} = \alpha_i \ \forall j$. Next, the M^+ model is analysed: The problem is shown to be polynomial when $\alpha_{ij} = \alpha_i$ $\forall j$ and the coefficients are increasing or decreasing in the order of the time windows;

complexity results for the remaining cases are also discussed. Finally, some conclusions and perspectives for future research are given.

Scheduling in M^+ and $M^\times$ according to a given sequence

We introduce an algorithm which solves the problem in both models. The following dominance property is used.

Theorem 1 *In models M^+ and $M^\times$, there exists an optimum schedule such that each job starts either at the beginning of a time window or at the completion time of the previously scheduled job.*

Proof Suppose there is a job j scheduled just after a job k. If $t_j = C_k$ the property is true, otherwise let i be the window in which j starts. If $t_j = d_i$, the property is true, else by setting $t_j = \max\{C_k, d_i\}$, the actual processing time of j does not change, neither in M^+ nor in $M^\times$, since j still starts in the same time window $[d_i, d_{i+1}[$. □

Without loss of generality we suppose now that the sequence is $(1, \ldots, n)$. Let $C(j,t)$ be the completion time of job j if it starts at time t (note that $C_j = C(j, t_j)$). The algorithm is the following:

Algorithm 1:

1. $t_1 = d_k$ with $C(1, d_k) = \min_{d_i} C(1, d_i)$

2. for $j = 2$ to n

$$t_j = \begin{cases} C_{j-1} & \text{if } C(j, C_{j-1}) \leq \min_{\{d_i : C_{j-1} \leq d_i\}} C(j, d_i) \\ d_k & \text{if } C(j, C_{j-1}) > C(j, d_k) = \min_{\{d_i : C_{j-1} \leq d_i\}} C(j, d_i) \end{cases}$$

Let t_j^* and C_j^* be the starting time and completion time of job j in an optimum schedule.

Theorem 2 *In models M^+ and $M^\times$, when the sequence of jobs is given, there exists an optimum schedule such that $C_j^* = C_j$ for every job j.*

Proof Consider the first job. By Theorem 1, $t_1^* = d_i$ for some i. So it cannot be completed before time $\min_{d_i} C(1, d_i) = C_1$. Therefore if $C_1^* > C_1$, Job 1 is shifted to the left in the optimum schedule, i.e $t_1^* = t_1$ and thus $C_1^* = C_1$.

Now, suppose $C_j^* = C_j$ for $1 \leq j \leq k$ and $k < n$. By Theorem 1, which applies to both models, $t_{k+1}^* = C_k^*$ or $t_{k+1}^* = d_i$ for some i such that $d_i \geq C_k^*$. Therefore job $k+1$ cannot be completed before time $\min\{C(k+1, C_k^*), \min_{\{d_i : C_k^* \leq d_i\}} C(k+1, d_i)\} = C_{k+1}$. Again, if $C_{k+1}^* > C_{k+1}$, the job is shifted to the left in the optimum schedule by setting $t_{k+1}^* = t_{k+1}$. So $C_{k+1}^* = C_{k+1}$ and, by induction, the proof is completed. □

Theorem 3 *When the sequence of jobs is given, the minimum makespan problem can be solved in models M^+ and $M^\times$ in time $\mathcal{O}(nw)$ by Algorithm 1.*

Proof - From Theorem 2, which applies to both models, it follows that $C_{\max} = C_{\max}^*$. Finally, because Step 2 of the algorithm is run $n-1$ times and there are $\mathcal{O}(w)$ comparisons each time, the total computation time is $\mathcal{O}(nw)$. □

The algorithm also solves the following special case, in both models.

Corollary 1 *When the sequence is not given, but the normal processing times are identical and the time window coefficients do not depend on the jobs, the minimum makespan problem can be solved in models M^+ and $M^\times$ in time $\mathcal{O}(nw)$*

Proof Since $x_j = x$, for some x, and $\alpha_{ij} = \alpha_i$, we have $p_j = \alpha_i x$ in $M^\times$ and $p_j = x + \alpha_i$ in M^+: The actual processing time of a job does not depend on the job. Consequently, Algorithm 1 can be used with any sequence to get an optimum schedule. □

Scheduling in $M^\times$ with two time windows

NP-hardness

We first prove that minimising the makespan is NP-hard if either $\alpha_{ij} = \alpha_i$ or $x_j = 1$, by reductions to the following problems:

- Problem Π_1:

  INSTANCE: A set of jobs $\mathcal{J}$, a sequence of two time windows $\mathcal{W} = (d_1, d_2, d_3)$, normal processing times x_j, coefficients α_{ij} such that $\alpha_{ij} = \alpha_i$, and an integer T.

  QUESTION: is there a schedule with makespan less than or equal to T?

- Problem Π_2:

  INSTANCE: A set of jobs $\mathcal{J}$, a sequence of two time windows $\mathcal{W} = (d_1, d_2, d_3)$, normal processing times $x_j = 1$, coefficients α_{ij}, and an integer T.

  QUESTION: is there a schedule with makespan less than or equal to T?

These problems are in NP since one can check in polynomial time if a schedule has a makespan less than or equal to T. We prove the NP-completeness of Π_1 and Π_2 by polynomial transformations from the PARTITION problem (Garey and Johnson [13]):

INSTANCE: Integers $b_1, \ldots, b_n$ such that $\sum_{i=1}^{n} b_i = 2B$.

QUESTION: Is there a subset $\mathcal{S} \subset \{1, \ldots, n\}$ such that $\sum_{i \in \mathcal{S}} b_i = B$?

Theorem 4 *Problem Π_1 is NP-complete.*

Proof With an instance of PARTITION is associated an instance of Π_1 as follows :

- $\mathcal{J} = \{1, \ldots, n\}$
- $d_1 = 0$, $d_2 = 2B$ and $d_3 = +\infty$
- $x_j = b_j$ for $1 \leq j \leq n$
- $\alpha_{1j} = 2$ and $\alpha_{2j} = 1$, for $1 \leq j \leq n$
- $T = 3B$

Figure 1 illustrates the transformation. The construction can be done in polynomial time. Now, suppose there is a "yes" answer for PARTITION. We get a solution of Π_1 as follows: jobs associated with $\mathcal{S}$ are scheduled first, followed by the remaining jobs. Since

jobs associated with $\mathcal{S}$ are completed by time d_2 exactly, the makespan is $2\sum_{j\in\mathcal{S}} b_j + \sum_{j\in\{1,\ldots,n\}-\mathcal{S}} bj = 2B + B = T$, and the schedule is a "yes" answer for Π_1. Conversely, suppose there is a "yes" answer for Π_1. Let $\mathcal{J}_1$ be the set of jobs starting before time d_2, $\mathcal{J}_2$ the set of jobs starting at or after time d_2, $x = \sum_{j\in\mathcal{J}_1} x_j$ and $y = \sum_{j\in\mathcal{J}_2} x_j$. First, note that $2\sum_{j\in\mathcal{J}_1} x_j + \sum_{j\in\mathcal{J}_2} x_j \leq T$, i.e $2x + y \leq 3B$, since $\sum_{j\in\mathcal{J}_1} p_j + \sum_{j\in\mathcal{J}_2} p_j \leq T$. Next, observe that $\sum_{j\in\mathcal{J}_1} x_j + \sum_{j\in\mathcal{J}_2} x_j = 2B$, i.e $x + y = 2B$, since $x_j = b_j$. Consequently, $y \geq B$. But $y \leq B$ because $\sum_{j\in\mathcal{J}_2} p_j \leq T - d_2 = B$. Therefore $x = y = B$ and the sets $\mathcal{J}_1$ and $\mathcal{J}_2$ define a "yes" answer for PARTITION. □

Theorem 5 *Problem Π_2 is NP-complete.*

Proof An instance of problem Π_2 is constructed by using the transformation in the proof of theorem 4 with $x_j = 1$, $\alpha_{1j} = 2b_j$ and $\alpha_{2j} = b_j$. The remainder of the proof is the same except that $x = \frac{1}{2}\sum_{j\in\mathcal{J}_1} \alpha_j$ and $y = \sum_{j\in\mathcal{J}_2} \alpha_j$. □

A pseudo-polynomial time algorithm for the $\alpha_{ij} = \alpha_i$ case

Lemma 1 *If $\alpha_1 < \alpha_2$, there exists an optimum schedule such that the last job to start in the first time window has the largest normal processing time.*

Proof Let m be a job with the largest normal processing time. Suppose we have an optimum schedule (without idle times since $\alpha_1 < \alpha_2$) such that the last job to start in the first time window is not m but a job j. We have two cases to consider according to the time window in which m starts:

1. If m starts in the first time window, the job can be interchanged with j since jobs starting in the same time window can be sequenced according to any order.

2. Otherwise, m starts in the second time window. As jobs starting in the same time window can be sequenced according to any order, m can be interchanged with the

first job starting in the second time window and consequently m starts at the time j is completed (remember that the schedule has no idle time). The completion time C_m is equal to $t_j + \alpha_1 x_j + \alpha_2 x_m$. If we interchange m and j, C'_j (the new completion time of j) is equal to $t_m + \alpha_1 x_m + \alpha_2 x_j = t_j + \alpha_1 x_m + \alpha_2 x_j$. Let us now compare C_m and C_j. We have $C'_j - C_m = (\alpha_1 - \alpha_2)(x_m - x_j)$. Since $\alpha_1 < \alpha_2$ and $x_m \geq x_j$, we get $C'_j \leq C_m$ and, as a result, the schedule remains optimum if the two jobs are interchanged. □

Theorem 6 *If there are two time windows and $\alpha_{ij} = \alpha_i$, the makespan minimisation problem, for model $M^{\times}$, can be solved in pseudo-polynomial time.*

Proof We shall prove that the problem can be modeled as a knapsack problem (see Garey and Johnson [13] for example), which is a well-known problem that can be solved in pseudo-polynomial time with very efficient algorithms (for up to several thousands of items, see Martello and Toth [14]). The knapsack problem with n items can be written as follows:

$$\begin{cases} \max \sum_j a_j y_j \\ \sum_j b_j y_j \leq d \\ y_j \in \{0,1\},\ \forall j \in \{1,\ldots,n\} \end{cases}$$

where a_j and b_j are respectively the cost and the weight of item j, and d is the capacity of the knapsack. The binary variable y_j is equal to 1 if item j is placed in the knapsack, and is equal to 0 otherwise. In our case, with each job j is associated a weight b_j equal to $\alpha_1 x_j$, and a boolean variable y_j such that $y_j = 1$ if and only if job j starts in the first time window. The values of a_j and d are defined as follows.

1. If $\alpha_1 < \alpha_2$, there is an optimum schedule without idle times and such that (by Lemma 1) a job with the largest normal processing time is the last one to start in the first time window. Let m be that job. The makespan of the schedule verifies $C_{\max} = \sum_{j \neq m} \alpha_1 x_j y_j + \alpha_1 x_m + \sum_{j \neq m} \alpha_2 x_j (1 - y_j)$. Moreover, jobs starting in the

first time window must complete before $d_2 - 1$ since since m is the last job to start in the first time window (recall that d_2 is the start time of the second time window). Hence we must have $\sum_{j \neq m} \alpha_1 x_j y_j \leq d_2 - 1$. Therefore, the makespan minimisation problem can be modeled as:

$$\begin{cases} min\ C_{\max} = \sum_{j \neq m} \alpha_1 x_j y_j + \alpha_1 x_m + \sum_{j \neq m} \alpha_2 x_j (1 - y_j) \\ \sum_{j \neq m} \alpha_1 x_j y_j \leq d_2 - 1 \\ y_j \in \{0, 1\},\ \forall j \in \mathcal{J} \end{cases}$$

Minimising $C_{\max}$ is equivalent to maximising $\sum_{j \neq m} (\alpha_2 - \alpha_1) x_j y_j$ since $\alpha_1 x_m + \sum_{j \neq m} \alpha_2 x_j$ is a constant. Hence, we have to solve a knapsack problem where $a_j = (\alpha_2 - \alpha_1) x_j$ and $d = d_2 - 1$:

$$\begin{cases} \max\ \sum_{j \neq m} (\alpha_2 - \alpha_1) x_j y_j \\ \sum_{j \neq m} \alpha_1 x_j y_j \leq a_2 - 1 \\ y_j \in \{0, 1\},\ \forall j \in \mathcal{J} \end{cases}$$

2. If $\alpha_1 > \alpha_2$, there may be an overlapping job in the optimum schedule. Hence, there are two cases:

   - If no job overlaps the two windows, the makespan depend on the jobs that start in the second time window, that is $C_{\max} = d_2 + \sum_j \alpha_2 x_j (1 - y_j)$. Moreover $\sum_j \alpha_1 x_j y_j \leq d_2$ since jobs starting in the first time window cannot be completed after time d_2 (otherwise there would be an overlapping job). Since $d_2 + \sum_j \alpha_2 x_j$ is a constant, the minimisation of the makespan is equivalent to the following knapsack problem where $a_j = \alpha_2 x_j$ and $d = d_2$:

   $$\begin{cases} \max\ \sum_j \alpha_2 x_j y_j \\ \sum_j \alpha_1 x_j y_j \leq d_2 \\ y_j \in \{0, 1\},\ \forall j \in \mathcal{J} \end{cases}$$

   - If one job (say m) overlaps the two windows, the makespan depends on the starting time t_m of the overlapping job m. It verifies $C_{\max} = t_m + \alpha_1 x_m +$

$\sum_{j \neq m} \alpha_2 x_j (1 - y_j)$ since job m is only followed by jobs that start in the second time window. Since job m starts in the first time window, it is completed before time $d_2 + \alpha_2 x_m$: Otherwise, we would get a schedule with no overlapping job by starting m in the second time window, at time d_2 exactly. Hence, $t_m + \alpha_1 x_m \leq d_2 + \alpha_2 x_m$, that is $t_m \leq d_2 + (\alpha_2 - \alpha_1) x_m$. Since the jobs that start in the first time window are completed before time t_m, $\sum_{j \neq m} \alpha_1 x_j y_j \leq d_2 + (\alpha_2 - \alpha_1) x_m$. Finally, minimising the makespan is equivalent to maximising $\sum_{j \neq m} \alpha_2 x_j y_j$ because $t_m + \alpha_1 x_m + \sum_{j \neq m} \alpha_2 x_j$ is a constant, and we get the following knapsack problem where $a_j = \alpha_2 x_j$ and $d = d_2 + (\alpha_2 - \alpha_1) x_m$:

$$\begin{cases} \max \sum_{j \neq m} \alpha_2 x_j y_j \\ \sum_{j \neq m} \alpha_1 x_j y_j \leq d_2 + (\alpha_2 - \alpha_1) x_m \\ y_j \in \{0, 1\}, \ \forall j \in \mathcal{J} \end{cases}$$

Therefore, we get a solution to the problem by solving $n+1$ knapsack problems. Indeed, we just have to solve the case with no overlapping job, and n cases with an overlapping one (one problem for each possible overlapping job). Then, the optimum schedule is the best schedule among these $n+1$ schedules. □

Scheduling in M^+

The $\alpha_{ij} = \alpha_i$ and increasing (or decreasing) α_i case

Lemma 2 *If $\alpha_i < \alpha_{i+1}$, there exists an optimum schedule such that the jobs are scheduled according to the non-decreasing normal processing time order.*

Proof Consider a schedule with two jobs j and j' such that $t_j < t_{j'}$ and $x_{j'} \leq x_j$. Let α and α' be the coefficients associated with the time windows in which j and j' start respectively. Let us denote by $\mathcal{S}$ the sequence of jobs scheduled during the time interval $[C_j, t_{j'}[$, i.e after j completes and before j' starts.

If j and j' are interchanged, the new schedule is such that j' starts at time $t'_{j'} = t_j$, followed by the jobs in $\mathcal{S}$ and then by job j, without idle times. To show that this interchange is always possible, we are going to prove that $C'_j \leq C_{j'}$, where C'_j is the completion time of j in the new schedule.

First, $C'_j = t'_{j'} + p'_{j'} + \sum_{k\in\mathcal{S}} p'_k + p'_j = t_j + x_{j'} + \alpha + \sum_{k\in\mathcal{S}} p'_k + x_j + \alpha''$, where α'' is the coefficient of the time window in which j starts. Then, because there may be idle times in the former schedule, $C_{j'} \geq t_j + x_j + \alpha + \sum_{k\in\mathcal{S}} p_k + x_{j'} + \alpha'$. Hence, the following inequality holds:

$$C'_j - C_{j'} \leq \sum_{k\in\mathcal{S}} (p'_k - p_k) + \alpha'' - \alpha'$$

Note that $C'_{j'} \leq C_j$ because $x_{j'} \leq x_j$. It follows that jobs in $\mathcal{S}$ start earlier in the new schedule and thus $\sum_{k\in\mathcal{S}} p'_k \leq \sum_{k\in\mathcal{S}} p_k$. Therefore, $\alpha'' \leq \alpha'$ since job j also starts earlier in the new schedule which implies that $C'_j \leq C_{j'}$. □

Lemma 3 *If $\alpha_i > \alpha_{i+1}$, there exists an optimum schedule such that the jobs are scheduled according to the non-increasing normal processing time order.*

Proof By using a similar interchange argument. □

Theorem 7 *If the model is M^+ and $\alpha_{ij} = \alpha_i$, the makespan minimisation problem can be solved in time:*

- $\mathcal{O}(n \log n)$, *if* $\alpha_i < \alpha_{i+1}$,
- $\mathcal{O}(n \log n + nw)$, *if* $\alpha_i > \alpha_{i+1}$.

Proof - If $\alpha_i < \alpha_{i+1}$, there exists an optimum schedule without idle times (it is always possible to shift a job to the left). Thus, to get an optimum schedule, it is enough to know how to sequence the jobs, that is, by Lemma 2, to sort the jobs according to the

non-decreasing normal processing time order. The time to solve the problem is then the time to sort n jobs, that is $\mathcal{O}(n \log n)$.

If $\alpha_i > \alpha_{i+1}$, there may be idle times in an optimum schedule. By Lemma 3 we know how to sequence the jobs. To get an optimum schedule, Algorithm 1 can be applied, which is defined for the case of a given sequence in Section . Sorting the jobs and applying the algorithm takes time $\mathcal{O}(n \log n)$ and $\mathcal{O}(nw)$ respectively, that is time $\mathcal{O}(\log n + nw)$. □

Complexity of the case with a fixed number of time windows

Let us first consider the case with two time windows. Alidaee and Womer [2] have shown that the makespan minimisation problem, for the piecewise linear model where $p_j = x_j$ if $t_j \leq d$ and $p_j = x_j + \alpha_j$ otherwise, can be solved in time $\mathcal{O}(n \log n)$ if the x_j's are identical. Mosheiov [9] proved that it is a binary NP-hard problem if we have arbitrary x_j's. We deduce from these two results that our problem (in which d_2 corresponds to $d + \epsilon$, where ϵ can be chosen as small as we want) can be solved in polynomial time if $\alpha_{1j} = 0$ and the x_j's are identical, but is NP-hard if $\alpha_{1j} = 0$ and the x_j's are arbitrary. Moreover, a consequence of Theorem 7 is that finding an optimum schedule if $\alpha_{ij} = \alpha_i$ is a polynomial time problem (with two time windows, either $\alpha_1 < \alpha_2$ or $\alpha_1 > \alpha_2$).

Therefore two open questions are (i) the complexity of the case where the α_{ij}'s are arbitrary but the x_j's are identical, and (ii) the complexity of the case where $\alpha_{ij} = \alpha_i$ and there are three time windows. In order to prove the NP-hardness of these two cases, the two following problems are considered:

- Problem Π_3:

  INSTANCE: A set of jobs $\mathcal{J}$, a sequence of three time windows $\mathcal{W} = (d_1, d_2, d_3, d_4)$, normal processing times x_j, coefficients α_{ij} such that $\alpha_{ij} = \alpha_i$ and an integer T.

  QUESTION: is there a schedule with makespan less than or equal to T?

- Problem Π_4:

  INSTANCE: A set of jobs $\mathcal{J}$, a sequence of two time windows $\mathcal{W} = (d_1, d_2, d_3)$, normal processing times $x_j = 1$, coefficients α_{ij} and an integer T.

  QUESTION: is there a schedule with makespan less than or equal to T?

It can be checked in polynomial time if a schedule has a makespan less than or equal to T, so the two problems are in NP. The NP-completeness of Π_3 and Π_4 is proved by polynomial transformations from a subcase of the PARTITION problem (Garey and Johnson [13]):

INSTANCE: Integers $b_1, \ldots, b_n$ such that $\sum_{i=1}^{n} b_i = 2B$.

QUESTION: Is there a subset $\mathcal{S} \subset \{1, \ldots, n\}$ such that $\sum_{i \in \mathcal{S}} b_i = B$ and $|\mathcal{S}| = \frac{n}{2}$?

Theorem 8 Π_3 *is NP-complete.*

Proof The transformation is the following (see Figure 2 for an illustration):

- $\mathcal{J} = \{1, \ldots, n+1\}$
- $x_j = b_j$, for $1 \leq j \leq n$
- $x_{n+1} = \frac{n}{2}B + 1$
- $d_1 = 0$, $d_2 = B + \frac{n}{2}B + 1$, $d_3 = 2B + nB + 1$, and $d_4 = +\infty$
- $T = 3B + \frac{3}{2}nB + 1$
- $\alpha_{1j} = \alpha_{3j} = B$, and $\alpha_{2j} = T$, for $1 \leq j \leq n+1$

The construction can be done in polynomial time. Now, suppose there is a "yes" answer for PARTITION. Jobs associated with $\mathcal{S}$ are first scheduled, followed by job $n+1$, then by the remaining jobs. Jobs associated with $\mathcal{S}$ are processed from time 0 to time $\sum_{j \in \mathcal{S}}(b_j + B) = B + \frac{n}{2}B$, since there are exactly $\frac{n}{2}$ jobs in $\mathcal{S}$. Hence, job $n+1$ starts at time $B + \frac{n}{2}B = d_2 - 1$ and is completed at time $d_2 - 1 + x_{n+1} + \alpha_{1n+1} = d_3$. Finally,

the remaining $\frac{n}{2}$ jobs are processed from time d_3 to time $d_3 + \sum_{j\in\{1,\ldots,n\}-\mathcal{S}}(b_j + B) = d_3 + B + \frac{n}{2}B = T$. Therefore the answer for Π_3 is also "yes". Conversely, let us consider a "yes" answer for Π_3. First, observe that no job can start within the second time window since $\alpha_{2j} = T$. Moreover, job $n+1$ cannot start in the third time window (otherwise $C_{n+1} > T$). In fact, job $n+1$ must start at time $d_2 - 1$: Indeed, if $t_{n+1} \leq d_2 - 2$, then at most $(d_2 - 2 - d_1) + (T - d_3) = 2B + nB - 1$ units of time are available for scheduling jobs 1 to n. But since these jobs start in time windows with coefficients equal to B, they need $\sum_{j=1}^{n}(x_j + B) = 2B + nB$ units of time to be processed: This is a contradiction. Suppose now that at least $\frac{n}{2} + 1$ jobs start before job $n+1$. Since their normal processing times are greater than or equal to 1 (recall the transformation), the last scheduled one is completed at least at time $\sum_{j=1}^{\frac{n}{2}+1}(1 + B) = (\frac{n}{2} + 1)(1 + B) > d_2$. But then job $n+1$ cannot start at time $d_2 - 1$. Similarly, there cannot be more than $\frac{n}{2}$ jobs starting after job $n+1$. Therefore, there are exactly $\frac{n}{2}$ jobs processed before job $n+1$ and $\frac{n}{2}$ jobs processed after. Let $\mathcal{J}_1$ and $\mathcal{J}_2$ be the set of jobs starting before and after job $n+1$, respectively. We know that $\sum_{j\in\mathcal{J}_1}(x_j + \alpha_{1j}) \leq d_2 - 1$, i.e $\frac{n}{2}B + \sum_{j\in\mathcal{J}_1} x_j \leq \frac{n}{2}B + B$. Similarly, we have $\frac{n}{2}B + \sum_{j\in\mathcal{J}_2} x_j \leq \frac{n}{2}B + B$. As a result, we get $\sum_{j\in\mathcal{J}_1} x_j = \sum_{j\in\mathcal{J}_2} x_j = B$, and the schedule defines a "yes" answer for PARTITION. □

Theorem 9 Π_4 *is NP-complete.*

Proof The following polynomial time transformation is used:

- $\mathcal{J} = \{1, \ldots, n\}$
- $x_j = 1$, for $1 \leq j \leq n$

Let $K = B + 1$, we set

- $d_1 = 0$, $d_2 = K(\frac{n}{2} + 1)B$, $d_3 = \infty$
- $\alpha_{1j} = K(b_j + B) - 1$ and $\alpha_{2j} = b_j + B - 1$, for $1 \leq j \leq n$.

- $T = K(\frac{n}{2}+1)B + (\frac{n}{2}+1)B$

Suppose there is a "yes" answer for PARTITION. Jobs in $\mathcal{S}$ are first scheduled from time 0 to time $\sum_{j\in\mathcal{S}} p_j = \sum_{j\in\mathcal{S}}(1+K(b_j+B)-1) = K(\frac{n}{2}+1)B = d_2$. The remaining jobs are then processed in the second time window and are completed by time $d_2+\sum_{j\in\{1,\ldots,n\}-\mathcal{S}}(1+b_j+B-1) = d_2+(\frac{n}{2}+1)B = T$: The schedule defines a "yes" answer for PARTITION. Now, suppose we have a "yes" answer for Π_4. First, note that no more than $\frac{n}{2}$ jobs can start in the second time window. Otherwise, as $p_j = b_j + B$ and $b_j \geq 1$, these jobs would be processed during at least $(\frac{n}{2}+1)(1+B)$ time units, which is not possible since the time window is of length $(\frac{n}{2}+1)B$ time units. Similarly, no more than $\frac{n}{2}$ jobs can start in the first time window. Otherwise, as $p_j = b_j + B$ and $b_j \geq 1$, these jobs would be processed during at least $(\frac{n}{2}+1)(K+KB) = K(\frac{n}{2}+1)B + (\frac{n}{2}+1)K > T$, since $K > B$. Hence $\frac{n}{2}$ jobs start in the first time window and $\frac{n}{2}$ in the second. Now, let $\mathcal{J}_1$ be the set of jobs starting before time d_2, $\mathcal{J}_2$ the set of jobs starting at or after time d_2, $x = \sum_{j\in\mathcal{J}_1} b_j$ and $y = \sum_{j\in\mathcal{J}_2} b_j$. First, note that $x + y = 2B$. Since $\sum_{j\in\mathcal{J}_2} p_j \leq T - d_2$, we also have $y \leq B$, and consequently $x \geq B$. Finally, because $\sum_{j\in\mathcal{J}_1} p_j + \sum_{j\in\mathcal{J}_2} p_j \leq T$, we have $\frac{n}{2}KB + Kx + \frac{n}{2}B + y \leq T$, i.e $Kx + y \leq KB + B$, which implies $x \leq B$. Therefore, $x = y = B$ and the sets $\mathcal{J}_1$ and $\mathcal{J}_2$ define a "yes" answer for PARTITION. □

Conclusion

This paper has introduced a new type of scheduling problems with time dependent processing times, where the time horizon is divided into time windows and the processing time of a job is associated with the time window in which the job starts. Two models are investigated. The first one is related to two other models proposed in the literature. The second model handles the case of jobs with processing times that are proportional to the availability of a ressource. Both models can also be used to approximate nonlinear

time dependent processing times by discretising the time horizon. The results presented in this paper are summarised in Table 1. Our current research aims at developing heuristic and exact procedures. In particular, two original integer programming models have been proposed, whose tight linear relaxations allow rather large instances to be solved using standard solvers.

References

1 Gawiejnowicz S (1996). A brief survey of continuous models of scheduling. *Found Comput Decis Sci* **21**: 81-100.

2 Alidaee B and Womer NK (1999). Scheduling with time dependent processing times: Review and extensions. *J Opl Res Soc* **50**: 711-720.

3 Cheng TCE, Ding Q and Lin BMT (2004). A concise survey of scheduling with time-dependent processing times. *Eur J Opl Res* **152**: 1-13.

4 Tanaev VS, Gordon VS and Shafransky YM (1980). *Scheduling Theory Single Stage Systems*. Nauka. (English translation 1994, Kluwer Academic Publisher).

5 Gupta JND and Gupta SK (1988). Single facility scheduling with nonlinear processing times. *Comput Ind Eng* **14**: 387-393.

6 Browne S and Yechiali U (1990). Scheduling deteriorating jobs on a single processor. *Opns Res* **38**: 495-498.

7 Gawiejnowicz S and Pankowska L (1995). Scheduling jobs with varying processing times. *Inform Process Lett* **54**: 175-178.

8 Sundararaghavan PS and Kunnathur AS (1994). Single machine scheduling with start time dependent processing times: Some solvable cases. *Eur J Opl Res* **78**: 394-403.

9 Mosheiov G (1995). Scheduling jobs with step-deterioration; minimizing makespan on a single and multi-machine. *Comput Ind Eng* **28**: 869-879.

10 Kononov AV (1997). On scheduling of a single machine jobs with processing times nonlinear in time. In: Korshunov AD (ed). *Operations Research and Discrete Analysis*, Kluwer Academic Pubisher: New-York, pp 109-122.

11 Cai J-Y, Cai P and Zhu Y (1998). On a scheduling problem of time deteriorating jobs. *J of Complexity* **14**: 190-209.

12 Kubiak W and Van de Velde S (1998). Scheduling deteriorating jobs to minimize makespan. *Naval Res Logist* **45**: 511-523.

13 Garey MR and Johnson DS (1979). *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman: San Francisco.

14 Martello S and Toth P (1990). *Knapsack Problems: Algorithms and Computer Implementations*. Wiley: Chichester.

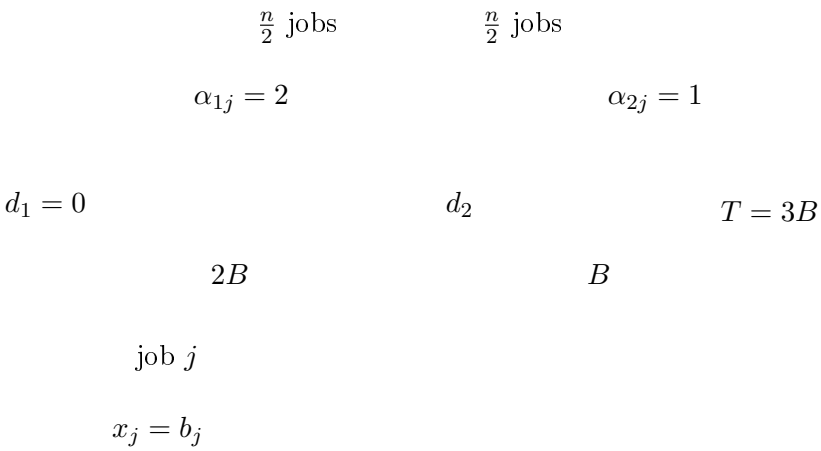

Figure 1: polynomial transformation for Theorem 4.

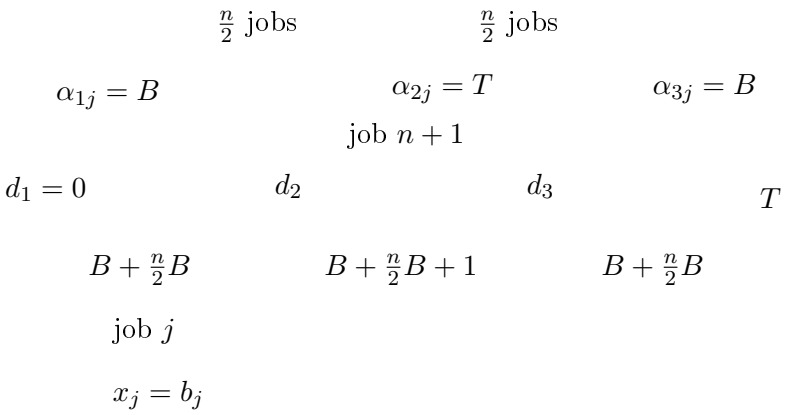

Figure 2: polynomial transformation for Theorem 8.

	$M^{\times}$	M^{+}
Given sequence or $\alpha_{ij} = \alpha_i$ and $x_j = x$	$\mathcal{O}(nw)$	$\mathcal{O}(nw)$
$\alpha_{ij} = \alpha_i$	if 2 windows : NP-hard but can be solved in pseudo-polynomial time	if $\alpha_i < \alpha_{i+1} : \mathcal{O}(n \log n)$ if $\alpha_i > \alpha_{i+1} : \mathcal{O}(n \log n + nw)$ else NP-hard if 3 windows
$x_j = 1$	NP-hard if 2 windows	NP-hard with 2 windows
2 windows and $\alpha_{1j} = 0$	Open	if $x_j = x : \mathcal{O}(n \log n)$ (Alidaee and Womer [2]) else NP-hard (Mosheiov [9])

Table 1: summary of computational complexity results.