Reactive and proactive single-machine scheduling to maintain a maximum number of starting times

Philippe Chrétienne

Reactive and proactive single-machine scheduling to maintain a maximum number of starting times

Philippe Chrétienne*

November 24, 2017

Abstract

This paper considers, in the single-machine scheduling context, the reactive and proactive problems arising when, due to unpredictable events between the time a baseline scheduled has been planned and the time the schedule must be implemented, the job durations may have increased so that the baseline schedule is no longer feasible. In the reactive case, the baseline schedule is known, the real job durations are known and we search for a schedule of the real instance that maximizes the number of jobs started at the same date in both schedules, this maximum being called the reactive gain. We show that, in the non-preemptive case, the corresponding decision problem is NP-complete in the strong sense while in the discrete preemptive case, it can be polynomially solved. In the proactive case, the real job durations are only known to belong to an uncertainty domain and we search for a baseline schedule that maximizes the worst reactive gain over the uncertainty domain. We show that the corresponding decision problem is NP-complete in the non-preemptive case while it is quite easy in the discrete preemptive case.

1 Introduction

Most often, when a scheduling problem has to be solved, a significative delay occurs between the time when a planned schedule is chosen and the time when it must be implemented. A usual consequence of this delay is that, for unpredictable reasons, the durations of some jobs have changed (most often increased), what makes the planned schedule no longer feasible.

*Sorbonne Universités, Université Pierre et Marie Curie, LIP6 CNRS UMR 7606,4 place Jussieu 75005 Paris, France, philippe.chretienne@lip6.fr

A *reactive problem* has thus to be solved when the real durations of the jobs are known. This problem is to determine a *definitive schedule*, that is to say a schedule of the real instance, that minimizes a cost that measures the gap between the planned schedule and the definitive schedule. At this level, this problem falls in the field of robust optimization [4] and has been widely studied in the scheduling context (see for example the survey [3]).

This minimum cost, which we call the *reactive cost*, depends on the planned schedule and on the variations of the durations. A cost measure that has been already studied is the sum of the absolute deviations of the corresponding starting times [2].

An other approach [6] comes from an application of the 2-stage robust LP model with right-hand-side uncertainty to robust PERT scheduling where the problem is to determine the minimum makespan that can be achieved for any realization of the job durations in a given uncertainty set.

However, in many applications, the cost is mainly impacted by the number of jobs that are not scheduled at the same date in the planned schedule and in the definitive schedule. The problem is then to find a definitive schedule such that the number of jobs scheduled at the same date in both schedules is maximum (or equivalently the reactive cost is minimum). It has been shown in [1] that this problem can be polynomially solved for CPM scheduling instances and when the real durations are longer that those of the planned instance.

The *proactive problem* takes place when a schedule has to be planned. At that time, just an estimation of the job durations, called the *planned durations*, is known and it is only assumed that the future variations of the durations belong to a known *uncertainty domain*. To each planned schedule corresponds the maximum reactive cost of this planned schedule over the uncertainty domain. The proactive problem is then to find a planned schedule that minimizes the maximum reactive cost.

In this paper, we study the reactive and proactive problems for the basic single-machine scheduling framework where independent jobs have to be processed on a single machine and all jobs must be completed before a given deadline. The reactive problem is called MAXANCHOR since it consists in maximizing an "anchorage level" defined by the number of jobs scheduled at the same date between the baseline schedule and the definitive schedule.

Section 2 deals with the reactive problem. In subsection 2.1, we give complexity results concerning the non-premptive case. We first study the so-called COMPATIBILITY problem where it must be decided if the jobs of a given subset of the real instance may be processed at their planned starting times. We show that COMPATIBILITY is NP-complete in the strong sense

and that even the special case COMPATIBILITY(1), where a single job has to be processed at its planned starting time is NP-complete. Coming back to our optimization problem MAXANCHOR where we search for a compatible subset with a maximum cardinality, we show that the decision problem corresponding to MAXANCHOR is NP-complete in the strong sense. In subsection 2.2, we study the discrete-preemptive variants called respectively DPR-COMPATIBILITY and DPR-MAXANCHOR where the processing of a job may be splitted into more than one interval with integer ends. We show that DPR-COMPATIBILITY may be solved in $O(n)$ time and we provide an $O(n^2)$ dynamic programming algorithm solving DPR-MAXANCHOR.

Section 3 deals with the proactive problem. We consider the special case when the perturbation of each job duration belongs to a known time interval. We show that finding a proactive schedule that minimizes the maximum reactive cost over the uncertainty domain, is an NP-complete problem. Instead, in the discrete preemptive case, we show that finding such a schedule is a quite easy problem.

2 The reactive problem

We assume that a given schedule $x = (x_1, \cdots, x_n)$ has been predetermined to execute non preemptively, on a single machine M, the jobs $J_1, \cdots, J_n$. Job J_i has a positive integer duration p_i and we assume without loss of generality that $x_1 < \cdots < x_n$. However, at the time when the schedule x must be actually implemented, it occurs that the real duration of job J_i is no longer p_i but $p_i + \delta_i$ where δ_i is a non-negative integer. Moreover, it is assumed that a common deadline D is imposed to the jobs of the real instance. The problem MAXANCHOR is then to find a schedule $y = (y_1, \cdots, y_n)$ of the real instance such that the number of jobs such that $x_i = y_i$, is maximum. These jobs are called the *on-time* jobs of y. An instance of MAXANCHOR is thus denoted by (J, x, p, δ, D) where $J = \{J_1, \cdots, J_n\}$.

It is interesting to note that the variant of MAXANCHOR where the deadline constraint is removed is polynomial since it is equivalent to the search of a maximum stable set in the interval graph (J, E) where there is an edge $e = \{i, j\}$ if the length of $[x_i; x_i + p_i + \delta_i] \cap [x_j; x_j + p_j + \delta_j]$ is positive [5].

We will first consider the COMPATIBILITY problem where, given a subset $H = \{h_1, \cdots, h_q\}$ of J, we have to decide whether there is a sched-

ule of the real instance such the jobs of H are scheduled at their planned starting time x_i. An instance of COMPATIBILITY will be denoted by (J, H, x, p, δ, D).

2.1 The non-preemptive case

Using a pseudopolynomial reduction from 3-PARTITION, we show that

Property 1. *COMPATIBILITY is NP-complete in the strong sense.*

Proof. Let (A, s) be an instance of 3-PARTITION, where $A = \{a_1, \cdots, a_{3m}\}$, $s : A \mapsto \mathbb{N}$ is such that $\sum_{i=1}^{3m} s(a_i) = mB$ and $\forall i \in \{1, \cdots, 3m\}, B/4 < s(a_i) < B/2$. Note that we necessarily have $B \geq 3$.
The corresponding instance $\mathcal{C}$ of COMPATIBILITY is as follows: for every a_i, there is a job J_i such that $p(J_i) = 1$, $\delta(J_i) = s(a_i) - 1$ and $x(J_i) = q(B+1)+r$ where q and r are such that $i-1 = 3q+r$ with $0 \leq r < 3$ (see Figure 1). $\mathcal{C}$ has also $m-1$ "blocking jobs" $K_1, \cdots, K_{m-1}$ such that $p(K_j) = 1$, $\delta(K_j) = 0$ and $x(K_j) = jB + j - 1$. Finally we have $D = mB + m - 1$ and $H = \{K_1, \cdots, K_{m-1}\}$.
Assume that the partition $\hat{A}_1, \cdots, \hat{A}_m$ of A is a solution of the instance (A, s). Then, since $B \geq 3$, the upper schedule of Figure 1 is feasible with respect to the planned durations of $\mathcal{C}$. Moreover the lower schedule of Figure 1 is feasible with respect to the real durations of $\mathcal{C}$ and H is a compatible subset of that schedule.

Assume now that H is a compatible subset of a schedule S of $\mathcal{C}$. Every blocking job K_j is scheduled in the interval $[jB; jB + 1]$. Since D is equal to the sum of the real durations of all jobs of $\mathcal{C}$, S has no idle time and we can conclude that the m subsets of jobs scheduled by S in the intervals $[j(B+1); j(B+1)+B]$ for all $j \in \{0, \cdots, m-1\}$ make a solution of (A, s). □

The problem COMPATIBILITY(1) is the special case of COMPATIBILITY when H is a singleton. Again, using a polynomial reduction from PARTITION, we show that

Property 2. *COMPATIBILITY(1) is NP-complete.*

Proof. Let (A, s) be an instance of PARTITION, where $A = \{a_1, \cdots, a_n\}$, $s : A \mapsto \mathbb{N}$ is such that $s(a_i) \geq 2$ for every $i \in \{1, \cdots, n\}$ and $\sum_{i=1}^{m} s(a_i) = 2B$. It is easy to see that this special case of PARTITION is also NP-complete.
The corresponding instance $\mathcal{C}$ of COMPATIBILITY(1) is as follows: for

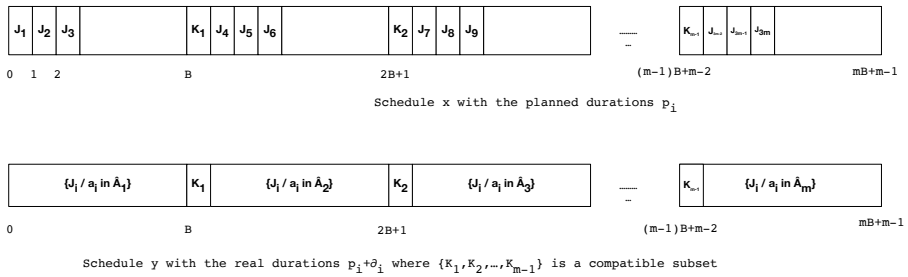

Figure 1: 3-PARTITION$\prec$COMPATIBILITY

every a_i, there is a job J_i such that $p(J_i) = 1$, $\delta(J_i) = s(a_i) - 1$ and $x(J_i) = i - 1$. $\mathcal{C}$ has also one "blocking job" K_0 such that $p(K_0) = 1$, $\delta(K_0) = 0$ and $x(K_0) = B$. Since $n \leq B$, the values $x(J_i) = i - 1, i \in \{1, \cdots, n\}$ make a feasible planned schedule. Finally, we let $D = 2B + 1$ and $H = \{K_0\}$.
Assume that the partition $\hat{A}_1, \hat{A}_2$ of A is a solution of the instance (A, s). Then the upper schedule of Figure 2 is feasible with respect to the planned durations of $\mathcal{C}$. Moreover the lower schedule of Figure 2 is feasible with respect to the real durations of $\mathcal{C}$ and H is a compatible subset of that schedule.
Assume now that H is a compatible subset of a schedule S of $\mathcal{C}$. The blocking job K_0 is scheduled in the interval $[B; B + 1]$. Since D is equal to the sum of the real durations of all jobs of $\mathcal{C}$, S has no idle time and we can conclude that the two subsets of jobs scheduled by S in the intervals $[0; B]$ and $[B + 1; D]$ make a solution of (A, s). □

The decision version D-MAXANCHOR of the problem MAXANCHOR is to decide, given an integer $K \leq n$, if there is a compatible subset of jobs with cardinality larger than or equal to K. An instance of D-MAXANCHOR is thus denoted by (J, x, p, δ, D, K). Using a reduction with nearly the same structure than that used for Property 1, we show that

Property 3. *D-MAXANCHOR is NP-complete in the strong sense.*

Proof. Let us first denote by 3-PARTITION$((3m + 3)s)$ the following variant of 3-PARTITION. An instance (A, s, σ) of 3-PARTITION$((3m + 3)s)$ is defined by:

- $A = \{a_1, \cdots, a_{3m}\}$,

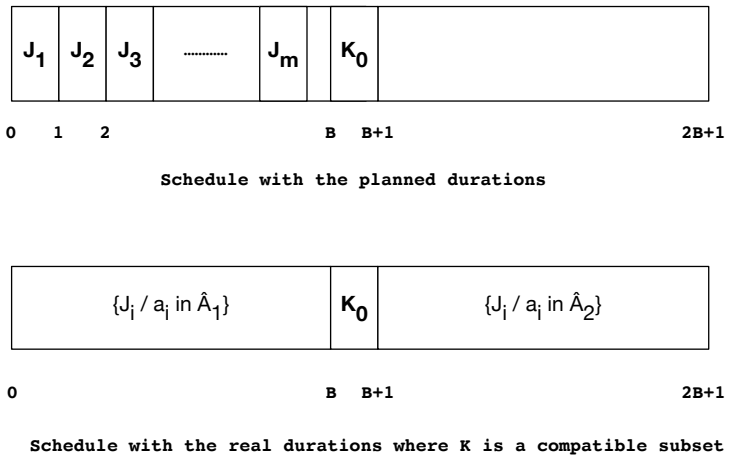

Figure 2: PARTITION$\prec$COMPATIBILITY(1)

- $s : A \mapsto \mathbb{N}$, such that $\sum_{i=1}^{3m} s(a_i) = mB$ and $\forall i \in \{1, \cdots, 3m\}, B/4) < s(a_i) < B/2)$
- $\forall i \in \{1, \cdots, 3m\}, \sigma(a_i) = (3m+3)s(a_i)$.

It must be decided whether there is a partition of A each class of which has size $(3m+3)B$ with respect to the size function σ. It is easy to see that 3-PARTITION$((3m+3)s)$ is also NP-complete in the strong sense.

Let (A, s, σ) be an instance of 3-PARTITION$((3m+3)s)$ and let $B' = (3m+3)B$. The corresponding instance of D-MAXANCHOR is as follows.

- For every a_i, there is a job J_i such that $p(J_i) = 1$, $\delta(J_i) = \sigma(a_i) - 1$ and $x(J_i) = i - 1$ (see Figure 3),
- There are also $m-1$ "blocking jobs" $K_1, \cdots, K_{m-1}$ such that $p(K_j) = 1$, $\delta(K_j) = 0$ and $x(K_j) = jB' + j - 1$.
- $D = mB' + m - 1$,
- $K = m$.

Assume that the partition $\hat{A}_1, \cdots, \hat{A}_m$ of A is a solution of the instance (A, s, σ). Without loss of generality, we can assume that $a_1 \in \hat{A}_1$. Since $B' \geq 3m$, the upper schedule x of Figure 3 is feasible with respect to the planned durations $p(J_i)$. In the lower schedule y of Figure 3, the jobs corresponding

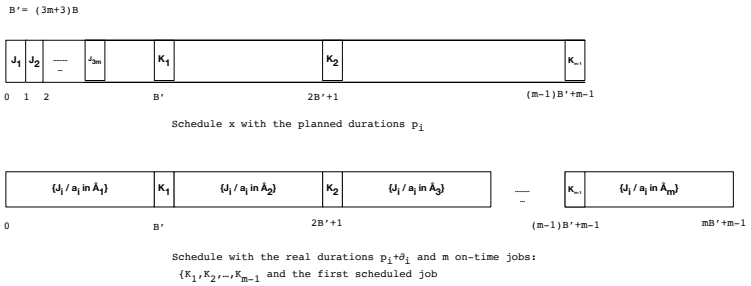

Figure 3: 3-PARTITION$((3m+3)s) \prec$D-MAXANCHOR

to $\hat{A}_1$ are scheduled in $[0; B']$ and J_1 is the first scheduled job in this interval. The jobs corresponding to $\hat{A}_j, j \in \{2, \cdots, m\}$ are scheduled in the interval $[(j-1)B' + j - 1; jB' + j - 1]$. Since $\sigma(a_1) > 3m$, J_1 is the only on-time job among the jobs $J_1, \cdots, J_{3m}$. So y is a feasible schedule and the number of on-time jobs in y is equal to m.

Assume now that y is a schedule of the instance of D-MAXANCHOR with at least m *on-time* jobs. Since the size of each item is greater than $3m$, at most one job in $\{J_1, \cdots, J_{3m}\}$ can be on-time. Thus the $m-1$ blocking jobs are on-time. Since D is equal to the sum of the processing times of all jobs, the m subsets of A associated with the jobs scheduled by y in the intervals $[(j-1)B' + j - 1; jB' + j - 1], j \in \{1, \cdots, m\}$ make a solution of the instance (A, s, σ). □

2.2 The discrete preemptive case

In the discrete-preemptive versions of COMPATIBILITY and MAXANCHOR, which are respectively denoted by DPR-COMPATIBILITY and DPR-MAXANCHOR, *the first time unit* during which every *on-time* job i is run must be the interval $[x_i; x_i + 1]$. Note also that in this context, discrete preemption is allowed for the planned schedule x. From now on, to simplify the presentation, the job J_i will be represented by its index i, $J = \{1, \cdots, n\}$, and we recall that, without loss of generality, it is assumed that $x_1 < \cdots < x_n$.

2.2.1 A linear-time algorithm for DPR-COMPATIBILITY

Let $I = (J, H, x, p, \delta, D)$ be an instance of DPR-COMPATIBILITY and let $\hat{I} = (J, H, x, p, \delta)$ be the corresponding instance when the deadline constraint

is removed. Also assume without loss of generality that $H = \{h_1, \cdots, h_q\}$ and $h_1 < \cdots < h_q$. We show that the algorithm $DPR.COMP(J, H, x, p, \delta)$ described below, where $[t]$ denotes the time-unit $[t; t+1]$), provides a schedule of (J, H, x, p, δ) with minimum makespan.

procedure $DPR.COMP(J, H, x, p, \delta)$;
(1) For each $k \in \{1, \cdots, q\}$, assign time-unit $[x_{h_k}]$ to the job h_k;
(2) For $k = 1$ to q
(2) assign the $p_{h_k} + \delta_{h_k} - 1$ first idle time-units $[t]$ with $t \geq x_{h_k}$ to job h_k;
(3) While not all jobs are completed
(3) assign the first idle time-unit to an uncompleted job.

In part (1), the jobs of H have to start exactly at their starting time in the given schedule x. In part (2), the assignment of the jobs of H is completed by scheduling them entirely in the order of the schedule and as early as possible but after their starting times. The schedule $S_2(H)$ of the jobs of H that we get at the end of part (2) will be shown to have a minimum makespan denoted by $M(S_2(H))$. In part (3), the schedule $S_2(H)$ is completed by scheduling the other jobs as early as possible in the idle periods of $S_2(H)$ for obtaining a schedule $S_3(H)$ which will also be shown to have a minimum makespan denoted by $M(S_3(H))$.

Figure 4 shows the schedule we get for the instance given by the following array and the compatible set $H = \{1, 3, 4, 5, 7\}$.

i	1	2	3	4	5	6	7
p_i	1	1	2	1	1	2	1
δ_i	2	2	1	1	1	2	1
x_i	0	1	7	11	13	16	18

It is easy to see that this algorithm, which may be implemented in linear time, finds a schedule such that each job of H is on-time. We now give some notations that are useful to show some properties of the schedules issued from steps 2 and 3 of $DPR.COMP(J, H, x, p, \delta)$.
The minimum makespan over the schedules of $\hat{I}$ is denoted by $\mu(J, H)$. The special instance (H, H, x, p, δ) (where x, p and δ are, in this case, the restrictions of x, p and δ to H) when all jobs must be on-time is denoted by $\hat{I}(H)$. The schedules provided by $DPR.COMP(J, H, x, p, \delta)$ satisfy the following properties:

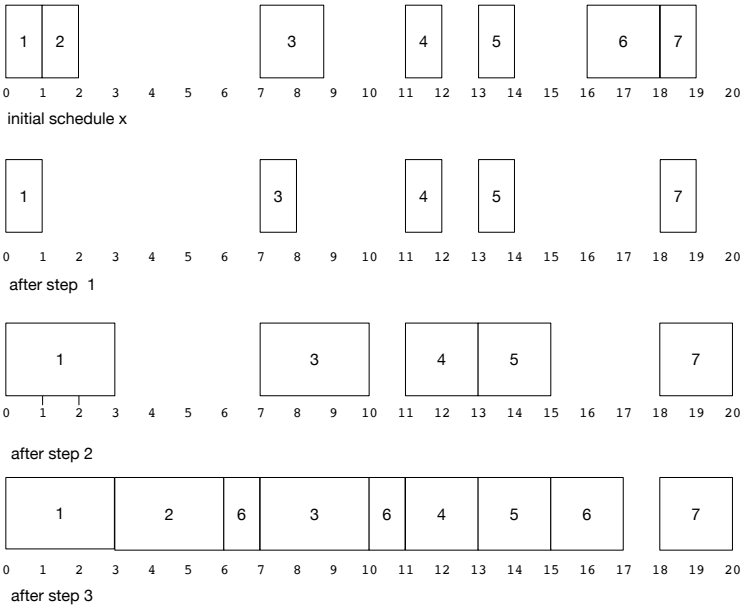

Figure 4: the 3 steps of $DPR.COMP$

Property 4. *The schedules $S_2(H)$ and $S_3(H)$ are such that*

- $M(S_2(H)) = \mu(H,H) = \max_{r\in\{1,\cdots,q\}}(x_{h_r} + \sum_{s=r}^{q}(p_{h_s} + \delta_{h_s}))$,
- $M(S_3(H) = \mu(J,H) = \max(\mu(H,H), \sum_{i=1}^{n}(p_i + \delta_i))$.

Proof. The schedule $S_2(H)$ is made of a sequence of intervals $I_1, \cdots, I_r$ where $I_k = [u_k; v_k]$ such that

- for $k \in \{1, \cdots, r-1\}$, $v_k < u_{k+1}$;
- $u_1 = x_{h_1}$ and $(u_1, \cdots, u_r)$ is a subsequence of $(x_{h_1}, \cdots, x_{h_q})$;
- for $k \in \{1, \cdots, r\}$, each time-unit of I_k is assigned to a job of H with x-values not less than u_k.

Assuming that $u_r = x_{h_r}$, we thus have $M(S_2(H)) = x_{h_r} + \sum_{s=r}^{q}(p_{h_s} + \delta_{h_s}))$. Since for any $j \in \{1, \cdots, q\}$, every job h_j is scheduled after the date x_{h_j}, we get $M(S_2(H)) \geq x_{h_j} + \sum_{s=j}^{q}(p_{h_s} + \delta_{h_s}))$. Moreover, since $\max_{r\in\{1,\cdots,q\}}(x_{h_r} + \sum_{s=r}^{q}(p_{h_s} + \delta_{h_s}))$ is obviously a lower bound of $\mu(H,H)$ and since the makespan of $S_2(H)$ is equal to that bound, we have $M(S_2(H)) = \mu(H,H)$.

It is clear, from step 3 of the algorithm, that if $M(S_2(H)) - \sum_{j=1}^{q}(p_{h_j} + \delta_{h_j}) \geq \sum_{i \in J \setminus H}(p_i + \delta_i)$, we have $M(S_3(H)) = M(S_2(H)) = \mu(H, H)$. Otherwise, there is not enough idle time-units in $S_2(H)$ to schedule the jobs of $J \setminus H$ and we have $M(S_3(H)) = \sum_{i=1}^{n}(p_i + \delta_i)$. We thus get $M(S_3(H)) = \max(\mu(H, H), \sum_{i=1}^{n}(p_i + \delta_i))$. Since $\max(\mu(H, H), \sum_{i=1}^{n}(p_i + \delta_i))$ is a lower bound of $\mu(J, H)$, we also have $M(S_3(H)) = \mu(J, H)$. □

The following property, which is a corollary of Property 4, will be useful to prove the algorithm of the next section.

Property 5. *Let H and K be two subsets of $\{1, \cdots, i-1\}$ such that $\mu(H, H) \leq \mu(K, K)$. Then we have $\mu(H \cup \{i\}, H \cup \{i\}) \leq \mu(K \cup \{i\}, K \cup \{i\})$.*

Proof. From Property 4, we get:

$$\mu(H \cup \{i\}, H \cup \{i\}) = \max(\mu(H, H), x_i) + p_i + \delta_i.$$

Since $\mu(H, H) \leq \mu(K, K)$, we also have

$$\mu(H \cup \{i\}, H \cup \{i\}) \leq \mu(K \cup \{i\}, K \cup \{i\}).$$

□

2.2.2 An $O(n^2)$ algorithm for DPR-MAXANCHOR

In order to solve DPR-MAXANCHOR, we consider the following dynamic programming algorithm:

procedure $DPR.MAXANCHOR(J, x, p, \delta)$;
(1) $m(1, 0) = p_1 + \delta_1$; $m(1, 1) = x_1 + p_1 + \delta_1$
(2) For $i = 2$ to n
(3) For $k = 2$ to $i - 1$
(4) $m^-(i, k) = \max(m(i-1, k), \sum_{j=1}^{i}(p_j + \delta_j))$;
(5) $m^+(i, k) = \max(x_i, m(i-1, k-1)) + p_i + \delta_i$;
(6) $m(i, k) = \min(m^-(i, k), m^+(i, k))$;
(7) $m(i, i) = \max(x_i, m(i-1, i-1)) + p_i + \delta_i$.

Let us introduce some more definitions to interpret the variables $m^-(i, k)$, $m^+(i, k)$ and $m(i, k)$. If $i \in \{1, \cdots, n\}$ and $H \subset \{1, \cdots, i\}$, the minimum makespan of a schedule of $\{1, \cdots, i\}$ such that H is compatible, is denoted by $M(i, H)$. Notice that $M(i, H)$ is a simpler notation for $\mu(\{1, \cdots, i\}, H)$.

For $i \in \{1, \cdots, n\}$ and $k \in \{0, \cdots, i\}$, we denote by $m(i,k)$ the minimum makespan of a schedule of $\{1, \cdots, i\}$ with k on-time jobs:

$$m(i,k) = \min\{M(i,H) | H \subset \{1, \cdots, i\}, |H| = k\}.$$

Let us define the bi-partition $(\mathcal{H}^-(i,k), \mathcal{H}^+(i,k))$ of the subsets of $\{1, \cdots, i\}$ with size k as follows:

- $\mathcal{H}^-(i,k) = \{H | H \subset \{1, \cdots, i-1\}, |H| = k\}$
- $\mathcal{H}^+(i,k) = \{H | H \subset \{1, \cdots, i\}, i \in H, |H| = k\}$

We then denote by $m^-(i,k)$ the value $\min\{M(i,H) | H \in \mathcal{H}^-(i,k)\}$ and by $m^+(i,k)$ the value $\min\{M(i,H) | H \in \mathcal{H}^+(i,k)\}$. From these definitions, we clearly have

$$m(i,k) = \min\{m^-(i,k), m^+(i,k)\}.$$

We first search for a recurrence formula satisfied by $m^-(i,k)$.
Let K be such that $m(i-1,k) = M(i-1,K)$. From Property 4, we have

$$m(i-1,k) = \max(\mu(K,K), \sum_{j=1}^{i-1}(p_j + \delta_j)).$$

For any $H \in \mathcal{H}^-(i,k)$, we have $\mu(K,K) \leq \mu(H,H)$ since otherwise, we would not have $m(i-1,k) = M(i-1,K)$. So, again from Property 4, we get

$$m^-(i,k) = \max(\mu(K,K), \sum_{j=1}^{i}(p_j + \delta_j)).$$

From these two equalities, we can conclude that

$$m^-(i,k) = \max(m(i-1,k), \sum_{j=1}^{i}(p_j + \delta_j)).$$

We now search for a recurrence formula satisfied by the function $m^+(i,k) = \min(M(i,H) | H \in \mathcal{H}^+(i,k)\})$. We consider two cases.

Assume first that $m(i-1,k-1) \leq x_i$. Let H be such that $M(i-1,H) = m(i-1,k-1)$ and let S be a schedule of the jobs $\{1, \cdots, i-1\}$ such that the jobs of H are on-time and whose makespan is $m(i-1,k-1)$. By adding to S the job i scheduled in the interval $[x_i; x_i + p_i + \delta_i]$, we get a schedule S' of the jobs $\{1, \cdots, i\}$ such that the jobs of $H \cup \{i\}$ are on-time and whose

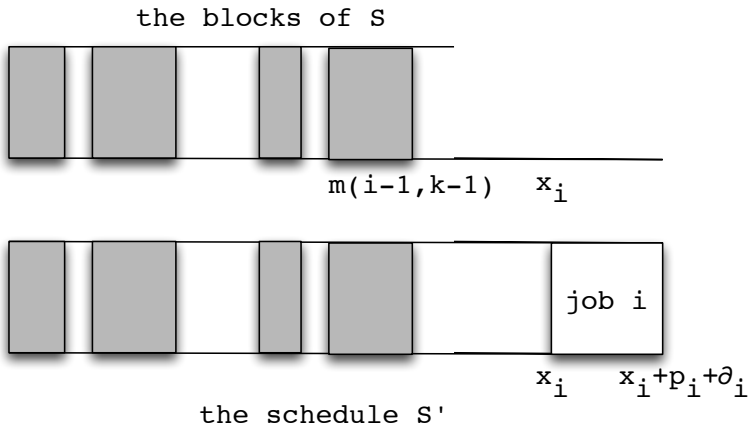

Figure 5: The schedules S and S'

makespan is $x_i + p_i + \delta_i$ (See Figure 5). Since $x_i + p_i + \delta_i$ is a lower bound of $m^+(i,k)$, we have

$$m^+(i,k) = x_i + p_i + \delta_i.$$

Assume now that $m(i-1,k-1) > x_i$. Let $K \subset \{1,\cdots,i-1\}$ be such that $m(i-1,k-1) = M(i-1,K)$ and let $K' = K \cup \{i\}$. We first show that there is a schedule of the jobs $\{1,\cdots,i\}$ with makespan $m(i-1,k-1)+p_i+\delta_i$ such that the jobs of K' are on-time.

Let S be a schedule of the jobs $\{1,\cdots,i-1\}$ with makespan $m(i-1,k-1)$ such that the jobs of K are on time. Consider the last block of S and let $[u;v]$ be the time interval of the last block of S (See Figure 6). We clearly have $u \leq x_{i-1}$ since otherwise all jobs of the last block of S could have been left-shifted by one time unit. Let r be the job of $\{1,\cdots,i-1\}$ scheduled in the time-unit $[x_i]$. By adding to S the job i scheduled in the interval $[m(i-1,k-1), m(i-1,k-1)+p_i+\delta_i]$, we get a schedule S' of the jobs $\{1,\cdots,i\}$ with makespan $m(i-1,k-1)+p_i+\delta_i$ such that the $k-1$ jobs of K are on-time. By exchanging in S' the jobs respectively scheduled at the time-units $[x_i]$ and $[m(i-1,k-1)+p_i+\delta_i-1]$ (See Figure 6), we get a schedule S'' of $\{1,\cdots,i\}$ with makespan $m(i-1,k-1)+p_i+\delta_i$ such that the k jobs of K' are on-time.

Assume now that there exists $\hat{H} \in \mathcal{H}^+(i,k)\}$ such that $M(i,\hat{H}) < m(i-1,k-1)+p_i+\delta_i$. Since job i is entirely scheduled after time x_i, we can remove job i and reschedule after time x_i and with no intermediate delay, all remaining jobs of $\{1,\cdots,i-1\}$ previously scheduled in S within the time

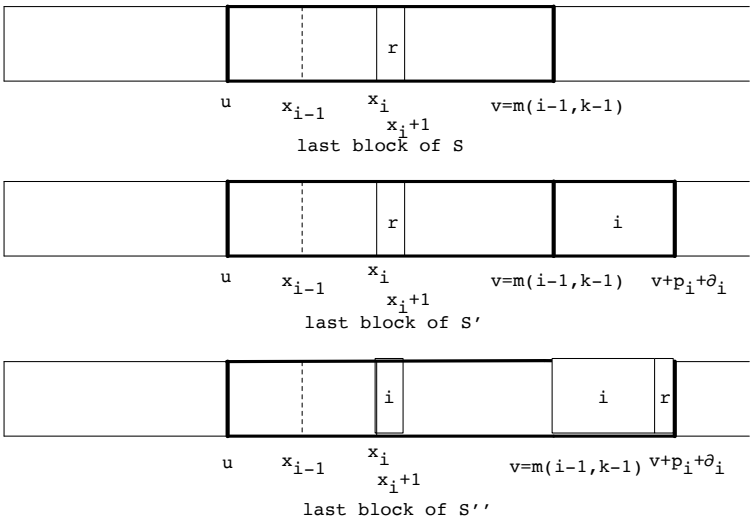

Figure 6: The schedules S, S' and S''

interval $[x_i; m(i-1,k-1)+p_i+\delta_i]$. We thus get a schedule of $\{1,\cdots,i-1\}$ with makespan less than $m(i-1,k-1)$ and $k-1$ on time jobs. A contradiction. We thus have $m^+(i,k) = m(i-1,k-1)+p_i+\delta_i$.

It finally comes from the study of the two cases that

$$m^+(i,k) = \max(x_i, m(i-1,k-1)) + p_i + \delta_i.$$

We can now conclude about the function $m(i,k)$.

- if $i > 1$ and $k \leq i-1$ we have

$$m(i,k) = \min \left\{ \begin{array}{l} \max(m(i-1,k), \sum_{j=1}^{i}(p_j+\delta_j) \\ \max(x_i, m(i-1,k-1)) + p_i + \delta_i \end{array} \right.$$

- if $k = i$, we have

$$m(i,i) = \max(x_i, m(i-1,i-1)) + p_i + \delta_i$$

- The initial values are the following

$$m(1,k) = \left\{ \begin{array}{ll} x_1 + p_1 + \delta_1 & \text{if } k = 1 \\ p_1 + \delta_1 & \text{if } k = 0 \end{array} \right.$$

So, given an instance (J, x, p, δ, D) of DPR-MAXANCHOR, the maximum number of on-time jobs is given by $\max(k \in \{1, \cdots, n\} | m(n, k) \leq D)$. Moreover, it is easy to see that the complexity of the dynamic programming algorithm is $O(n^2)$.

3 The proactive problem

In the proactive problem, the variations δ_j of the job durations are only known to belong to an uncertainty domain Δ. It is also assumed that, whatever the realization δ in Δ, the real instance $(J, p + \delta, D)$ has at least one schedule since, otherwise, it could happen that the real instance has no feasible schedule. Let $I = (J, p, d)$ be the initial instance, where d is the common deadline that must be satisfied by the jobs of J in a planned schedule. If x is a schedule of I and if $\delta \in \Delta$, we denote by $r(x, \delta)$ the *reactive gain* of the planned schedule x to the variations vector δ, that is to say the maximum number of jobs that may be scheduled on-time when the schedule x has been planned and δ is known. Given a planned schedule x of I, we denote by $R(x, \Delta)$ the value $\min_{\delta \in \Delta} r(x, \delta)$, which is the worst number of jobs that could be scheduled on-time due to uncertainty, if the planned schedule x is chosen. The proactive problem is finally to find a planned schedule x^* such that $R(x^*, \Delta)$ is maximum.
We consider in this paper the special case when each δ_j belongs to a known interval $[\delta_j^-; \delta_j^+]$ and we denote respectively by δ^+ and δ^- the corresponding vectors. In that case, the following monotonicity property will play a major role

Property 6. *Let x be a schedule of the instance (J, p, d). If $\delta, \delta' \in \Delta$, and $\delta' \leq \delta$, then $r(x, \delta') \geq r(x, \delta)$.*

Proof. Let $H \subset J$. If the instance (J, x, p, δ, D) of $MAXANCHOR$ has a schedule such that the jobs of H are on-time, then clearly the same schedule is also feasible for the instance (J, x, p, δ', D). □

3.1 The proactive problem for $MAXANCHOR$

From Property 6, we easily get that the problem of finding the best proactive schedule x^* is equivalent to the following problem : given the instances $I = (J, p, d)$ and $I^+ = (J, p + \delta^+, D)$ where $d \geq \sum_{i \in \{1, \cdots, n\}} p_i$ and $D \geq \sum_{i \in \{1, \cdots, n\}} (p_i + \delta_i^+)$, find a schedule x of I and a schedule y of I^+ such that the number of jobs $eq(x, y)$ with $x_i = y_i$ is maximum. We call

the decision version of this problem $PROMAXANCHOR$ and denote by $(J, p, \delta^+, d, D, k)$ the instance where it is asked if at least k jobs may be scheduled at the same date in the two schedules.
Let us denote by Π the special case of $PROMAXANCHOR$ when $d = \sum_{i\in\{1,\cdots,n\}} p_i$, $D = \sum_{i\in\{1,\cdots,n\}}(p_i + \delta_i^+)$ and $k = 2$. Before showing that Π is NP-complete, we prove the following property

Property 7. *Let $I = (J, p, \delta^+, d, D)$ an instance of Π. I has at least one solution (x, y) with $eq(x, y) \geq 2$ if and only if there are two subsets $K, L \subset J$ such that :*

- $K \cup L \subsetneq J$,
- $K \cap L \neq \emptyset$,
- $\sum_{i\in K} p_i = \sum_{i\in L}(p_i + \delta_i^+)$.

Proof. Assume there are two subsets $K, L \subset J$ satisfying the 3 conditions, let $i \in K \cap L$ and let $j \in J \setminus (K \cup L)$. Then the solution (x, y) shown by Figure 7 satisfies $eq(x, y) \geq 2$. Conversely, assume I has a solution (x, y) with $eq(x, y) \geq 2$, let J_i and J_j be two jobs such that $x_i = y_i$ and $x_j = y_j$ and assume without loss of generality that $x_i < x_j$. Let K be the set of jobs scheduled by x in the interval $[x_i; x_j]$ and let L be the set of jobs scheduled by y in the interval $[y_i; y_j]$. Then we clearly have $J_j \in J\setminus(K \cup L)$, $J_i \in K\cap L$ and $\sum_{i\in K} p_i = \sum_{i\in L}(p_i + \delta_i^+)$. □

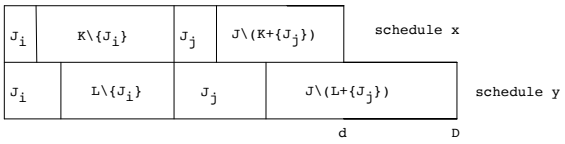

Figure 7: The schedules (x, y) corresponding to K, L.

Property 8. Π *is NP-complete.*

Proof. We show that $PARTITION$ polynomially reduces to Π.
Let (A, s) be an instance of $PARTITION$ where $A = \{a_1, \cdots, a_n\}$, $s : A \mapsto \mathbb{N}$ and $\sum_{i=1}^n s(a_i) = 2B$. The corresponding instance of Π is as follows:

- $J = \{J_1, \cdots, J_n\}$,

- $\forall i \in \{1, \cdots, n\}, p_i = s(a_i)$,
- $\forall i \in \{1, \cdots, n\}, \delta_i^+ = B - s(a_i)$.

We note that for that instance, we have $d = 2B$ and $D = nB$.
Assume first that (A_1, A_2) is a solution of the instance (A, s) of $PARTITION$ and let K be the subset of jobs corresponding to A_1, let J_i be a job of K, let $L = \{J_i\}$ and let J_j be one of the jobs associated with A_2. Then the schedules x and y shown on Figure 8 make a solution of the instance of Π such that $eq(x, y) \geq 2$.
Assume now that the instance of Π has a solution (x, y) such that $eq(x, y) \geq 2$. We know that there are two subsets $K, L \subset J$ such that the three conditions of Property 7 are satisfied. We get from the third condition that $\sum_{i \in K} p_i = |L| \times B$. Now from the first condition we need to have $|L| = 1$. So, if $A_1 = \{a_i | J_i \in K\}$, then $(A_1, A \setminus A_1)$ is a solution of the instance (A, s).

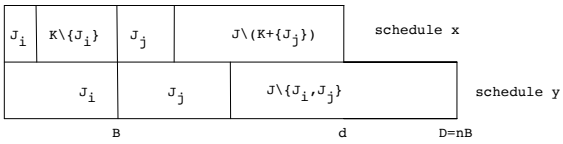

Figure 8: A solution (x, y) of the instance of Π.

□

Pointing out that, without loss of generality, the parameter d of an instance of $PROMAXANCHOR$ may be assumed to be at most $\sum_{i=1}^{n}(p_i + \delta_i^+)$, we get that the length of Π and the length of $PROMAXANCHOR$ are polynomially equivalent. So $PROMAXANCHOR$ is also an NP-complete problem.

3.2 The proactive problem for $DPR - MAXANCHOR$

As for the case of $MAXANCHOR$, we easily get from Property 6 that the problem of finding the best proactive schedule x^* is equivalent to the following problem : given the instances $I = (J, p, d)$ and $I^+ = (J, p + \delta^+, D)$ where $d \geq \sum_{i \in \{1, \cdots, n\}} p_i$ and $D \geq \sum_{i \in \{1, \cdots, n\}} (p_i + \delta_i^+)$, find a *discrete-preeemptive* schedule x of I and a *discrete-preemptive* schedule y of I^+ such that the number of jobs $eq(x, y)$ with $x_i = y_i$ is maximum. Consider the schedule x^* of I where, for each job J_i, the first time-unit assigned J_i is $[i]$ and where

the remaining $\sum_{i=1}^{n} p_i - n$ processing time-units of the jobs are assigned the the time interval $[n; \sum_{i=1}^{n} p_i]$ in an arbitrary order (See Figure 9). It is easy to see that the schedule provided by $DPR.COMP(J, J, x^*, p, \delta^+)$ has no idle time so that its makespan is $\sum_{i=1}^{n}(p_i + \delta_i)$. We thus have $R(x^*, \Delta) = r(x^*, \delta+) = n$ and conclude that x^* is an optimal proactive schedule.

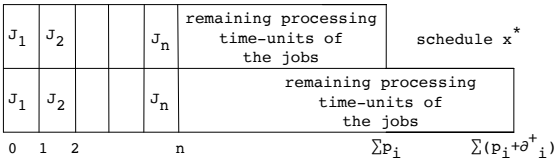

Figure 9: The optimal proactive schedule

4 Conclusion

The results presented in this paper make a first contribution to that new class of problems in the single-machine scheduling context. In the non preemptive case, solving efficiently the reactive and proactive problems makes an interesting future research direction. In the preemptive case, the complexity of the weighted version where a cost is due for each delayed job is still unknown. In both cases, the proactive problem should be studied for other kinds of uncertainty domains, such as the case when the perturbations are limited by a global budget constraint. Finally, the problem studied in this paper does not only concern scheduling problems but all combinatorial problems where a solution of a planned instance must be chosen before the real instance is actually known.

References

[1] Bendotti P., Chrétienne P., Fouilhoux P. and Quilliot A. *Anchored reactive and proactive solutions to the CPM scheduling problem.* European Journal of Operational Research, Feb 2017, DOI: 10.1016/j.ejor.2017.02.007.

[2] Herroelen, W and Leus, R. *The construction of stable project baseline schedules.* Technical Report Katholieke Universiteit Leuven, 2004.

[3] Herroelen, W., and Leus, R. (2004). Robust and reactive project scheduling: a review and classification of procedures *Int. j. prod. res.*, 42(8):1599-1620.

[4] Ben-Tal, A., El Ghaoui, L. and Nemirovski, A. (2009). Robust Optimization. *Princeton Series in Applied Mathematics, Princeton University Press*, 9-16.

[5] Fulkerson D.R., Gross O.A.(1965). Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, vol 15, 835-855.

[6] Minoux M.(2011) On 2-stage robust LP with RHS uncertainty: complexity results and applications. *Journal of Global Optimization*, vol 49, 521-537.