On maximizing the profit of a satellite launcher: selecting and scheduling tasks with time windows and setups

P. Baptiste *
CNRS LIX, Ecole Polytechnique, France

P. Chrétienne†
Universit Paris 6,CNRS LIP6, France

J. Meng-Grard‡
Universit Paris 6,CNRS LIP6, France

F. Sourd§
CNRS LIP6, France

November 23, 2007

Abstract

At regular times, a satellite launcher company has to plan the use of its launcher to get a maximum profit. In a more formal way, the problem consists in selecting and scheduling a subset of unit-length jobs constrained by capacitated time slots so that the overall cost is minimum. The data associated with each job are its weight, its time-window and its expected gain when it is performed. With each time slot are associated a setup cost and a capacity. The setup cost of a time slot is due when this time-slot is used to perform at least one job. Moreover the total weight of all jobs scheduled within a time slot is at most the time slot capacity. We first show that the general problem is hard and provide some easy special cases. We then propose a first dynamic-programming polynomial-time algorithm for the special case with unit weights. A second and more efficient dynamic proramming algorithm is also provided for the special case of unit weights and agreeable time windows. This last algorithm is finally improved for the special case of equal gains.

*Philippe.Baptiste@polytechnique.fr
†Philippe.Chretienne@lip6.fr
‡contact@jie.fr
§Francis.Sourd@lip6.fr

1 Introduction

The problem studied in this paper originates in maximizing the profit of a satellite launcher company. At a given decision time, the satellites to be launched have to be selected among all current demands of the customers, each launch has to be scheduled within a given time window and the launcher has a global capacity constraint. As observed in [2], this problem falls within a more generic class of scheduling problems that may be described as follows: given a set of unit-length jobs linked by compatibility constraints, where each job has a time-window and a realization profit and each time slot has a setup cost, one must schedule a subset of the jobs in order to get a maximum profit (or equivalently a minimum cost). In this paper, we consider the special case when the compatibility constraints come from a global capacity constraint: with each job is associated a positive weight (e.g. the satellite mass) and the sum of the weights of the jobs scheduled in a time slot must not exceed the overall capacity of the launcher.

Scheduling with rejection has already been studied in the literature. In the on-line context, each arriving job may be either rejected at a certain penalty or scheduled without any information about the future jobs. The objective is then to minimize the makespan plus the sum of the penalties of the rejected jobs. For the multiprocessor case, Bartal et al.[6] provide an on-line algorithm with competitive ratio $1 + \Phi$ (where Φ is the golden ratio) and show that it is the best possible competitive algorithm. For the same on-line problem but when preemption is allowed, a 2.388-competitive algorithm has been presented in [7]. More recently in [8], the ability to purchase a new machine with a certain machine cost when a new job arrives, has been added and an optimal on-line algorithm with competitive ratio of 2 has been provided.

In the off-line context, Engels et al. [5] combine penalties for each rejected job with the sum of the weighted completion times and describe several techniques for designing scheduling algorithms under this criterion. In [1], the authors study the combination of rejected job penalties with the sum of tardiness. For the multiprocessor preemptive case and the makespan minimization, in addition to a complete classification with respect to complexity and approximability, a 1.58-approximation algorithm for an arbitrary number of unrelated machines is given in [11]. In problems without deadlines, job rejection can be modeled by minimizing the number of tardy jobs: indeed a late job can be scheduled as late as necessary at no extra cost, what is equivalent to consider that the job is not selected. A survey of these problems is proposed in [4]

Scheduling with compatibility constraints has more recently been studied. Mutual exclusion scheduling concerns multiprocessor problems with unit-length-job problems where an incompatibility graph indicates which pairs of jobs cannot be simultaneously scheduled. Complexity results as well as polynomial special cases for interval graphs or related classes may be found in [9]. Parallel batch scheduling problems where a compatibility graph gives the pairs of jobs that may be put in the same batch also falls within this domain and give rise to difficult bounded vertex colouring problems if there is a batch capacity [10].
The originality of this paper is to study the combination of rejection, time windows and capacity constraints to schedule a set of unit length jobs with the additional features that a non-empty time slot incurs a specific setup cost and that a selected job induces a specific gain. Section 2 gives the notations of the problem. Section 2 studies the complexity of the general problem and presents some easy special cases. The two last sections concern the special case of equal weight jobs. In Section 3, a dynamic programming algorithm is given for arbitrary time windows. Section 4 presents a more efficient dynamic programming algorithm for the special case of agreeable time windows and a still more efficient variant for equal gains. The conclusion gives research directions for future works.

2 Problem definition and complexity

2.1 Problem definition

We consider a set $J = \{1; \cdots; n\}$ of n unit-length independent jobs each of which has an individual weight w_i (to be understood as the weight of an item in a knapsack problem). For each job i, it must be decided whether it will be processed or not. In the positive case, it must be decided in which time-slot of an a priori given time window $[r_i; d_i]$ (where r_i and d_i are non-negative integers) it must be performed. A positive *gain* g_i results from performing i and a positive *setup* cost K_t is due if at least one job is assigned to time slot $[t-1; t]$ (also denoted by t). Finally, the sum of the weights of the jobs performed in any time-slot must not exceed a given capacity C. So a generic instance of the problem will be denoted by $(n; w; C; r; d; g; K)$. Let $T = \bigcup_{j \in J} \{r_j + 1; \cdots; d_j\}$. A solution S of the problem is a pair $(E; s)$ where:

- $E \subseteq J$ is the set of selected jobs,
- $s : E \mapsto T$ indicates the time-slots assigned to the jobs of E,

- for any $j \in E$, $s(j)$ must be in $\{r_j + 1; \cdots; d_j\}$,
- for any time-slot t, the sum $\sum_{\{j|s(j)=t\}} w_j$ must be at most C.

The cost of the solution $(E; s)$ is $\sum_{t \in s(E)} K_t - \sum_{j \in E} g_j$ and the problem is to find a minimum-cost solution. It will be convenient to refer to this problem as problem Π. It will be assumed that $\bigcup_{j \in J}[r_j; d_j)$ constitutes a single interval since otherwise, the problem may be decomposed into smaller independent subproblems.

2.2 Complexity results

Theorem 1 *Problem* Π *is strongly NP-hard.*

Proof. — The following (pseudo polynomial) reduction of 3-PARTITION to Π shows the theorem. Assume that $(A; s)$ with $A = \{a_1; \cdots; a_{3m}\}$, $\sum_{i=1}^{3m} s(a_i) = mB$ and for any $i \in \{1; \cdots; 3m\} : \frac{B}{4} < s(a_i) < \frac{B}{2}$, is an instance of 3-PARTITION. The question is: is there a partition of A into m subsets $A_1; \cdots; A_m$ such that for any $k \in \{1; \cdots; m\}$, $\sum_{a_i \in A_K} s(a_i) = B$?
The corresponding instance $(n; w; C; r; d; g; K)$ of the decision version of Π is defined by: $n = 3m\,; \forall i \in \{1; \cdots; 3m\} : w_i = s(a_i); r_i = 0; d_i = m; g_i = 1\,; \forall t \in \{1; \cdots; m\} : K_t = 0\,; C = B$ and the question is whether there is a solution with profit at least $3m$. Clearly, if $(A; s)$ is a yes instance with solution $A_1; \cdots; A_m$, then by assigning the jobs associated with A_i to time slot i, we get a solution of the corresponding instance of Π with profit $3m$. Conversely, in a schedule of the instance of Π with profit $3m$, all the jobs need to be selected, which defines a feasible partition of the instance $(A; s)$.

■

The special case with $K_t = 0$, $g_i = 1$ and $(r_i; d_i) = (0; 2)$ is weakly NP-complete since it corresponds to the problem of finding the maximum number of weighted jobs that may be assigned to two time slots, each with a given capacity C and PARTITION polynomially reduces to that problem.
The single-machine special case defined by $C = 1$ and unit weights is easy. Since at most one job can be scheduled in any time slot, we simply have an assignment problem of the jobs to the time-slots where the cost to assign time slot t to job j is given by $K_t - g_j$.
Also the special case defined by unit weights and null setup costs is easy since we get the minimum-cost flow problem to assign jobs to capacitated time intervals so that the overall gain is maximum. Here, the time intervals are those delimited by the increasing sequence of the distinct values of the

release times and deadlines. The arcs from the source to the jobs have capacity one and a null cost; an arc from job i to time interval k (that only exists if job i may be executed in time interval k) has capacity one and cost $-g_i$; finally the arc from time interval k to the sink has capacity $l_k C$ where l_k is the length of time interval k.

3 The unit-weight case

From now on, we consider the special case $\Pi(1)$ of Π such that $w_j = 1$ for each job j. We first show that with a formulation similar to that used in [3], the problem may be solved in polynomial time by a dynamic programming algorithm. An instance of $\Pi(1)$ will be denoted by $(n; C; r; d; g; K)$. We will assume without loss of generality that the jobs are indexed according to the non-decreasing order of their deadlines,i.e: $d_1 \leq \cdots \leq d_n$. We also denote by $(a_0; \cdots; a_q)$ the strictly increasing sequence of the distinct release times and deadlines. The *band* B_k, $k \in \{1; \cdots; q\}$ is the interval $[a_{k-1}; a_k]$. The *capacity* C_k of the band B_k is defined as $C \cdot (a_k - a_{k-1})$. For any job j, $D(j) = \{b_j^-; \cdots; b_j^+\}$ is the set of the consecutive band indices of the interval $[r_j; d_j]$. Moreover if job j is selected, then $b(j)$ denotes the band index where j is processed. These notations are illustrated in Figure 1.

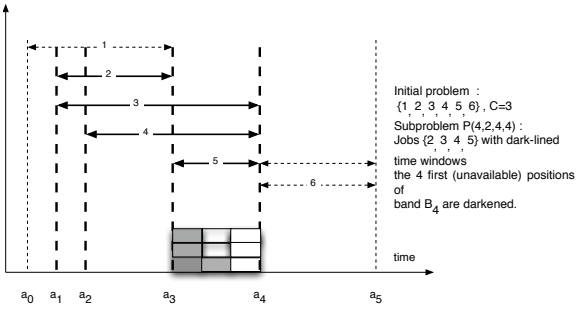

Figure 1: A schedule of an instance of $\Pi(1)$

Finally, as shown in Figure 2, each band B_k has C_k available *positions*. Without loss of generality, we will assume that the successive time slots of a band have non-decreasing setup costs. So the positions of band B_k are ordered as shown in Figure 2.
We now present a dominance property that will be the key of the dynamic programming algorithm solving problem $\Pi(1)$.

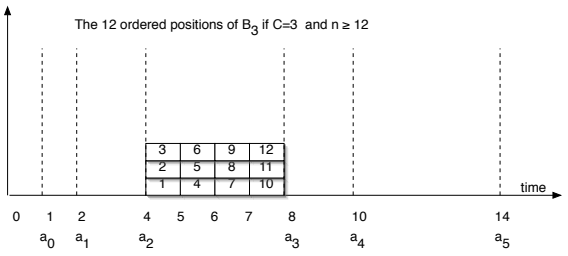

Figure 2: The ordered positions of a band

Lemma 1 *Let* $I = (n; C; r; d; g; K)$ *be an instance of* $\Pi(1)$. *There is an optimal solution* $(E; s)$ *of* I *such that:* $i; j \in E$, $i < j$ *and* $r_i \leq s(j)$ *implies* $s(i) \leq s(j)$.

Proof. — Assume that, as illustrated in Figure 3, there is an optimal solution $(E; s)$ such that $i < j$, $r_i \leq s(j)$ and $s(i) > s(j)$. By exchanging the time slots assigned to jobs i and j, we still get a feasible solution $(E; s')$ since $r_i \leq s(j) = s'(i) < s(i) < d_i$ and $r_j \leq s(j) < s'(j) = s(i) < d_i \leq d_j$ with the same cost as $(E; s)$. Iterating this process, we get an optimal solution satisfying the lemma. ∎

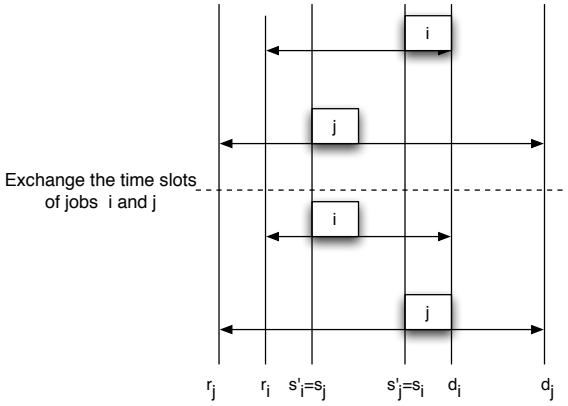

Figure 3: The main dominance rule

Note that the schedule shown in Figure 1 is not dominant since $r_4 < s(5)$ and $s(4) > s(5)$.
In our algorithm, the jobs will be assigned to the band indices and not to

the time slots, therefore we reformulate Lemma 1 as follows (where $a_{b(j)-1}$ is the starting time of the band where job j is processed):

Corollary 1 *Let $I = (n; C; r; d; g; K)$ be an instance of $\Pi(1)$. There is an optimal solution $(E; s)$ of I such that: $i; j \in E$, $i < j$ and $r_i \leq a_{b(j)-1}$ implies $b(i) \leq b(j)$.*

Proof. — Assume that there is an optimal solution $(E; s)$ such that $i < j$, $r_i \leq a_{b(j)-1}$ and $b(i) > b(j)$. By exchanging the time slots assigned to jobs i and j, we still get a feasible solution $(E; s')$ since $r_i \leq a_{b(j)-1} \leq s(j) = s'(i) < s(i) < d_i$ and $r_j \leq s(j) < s'(j) = s(i) < d_i \leq d_j$ with the same cost as $(E; s)$.

■

Let us now consider the *single-band problem* where all the jobs have the same time window (without loss of generality the first T time slots). We have $n = Cb + r$ for some natural numbers b and r such that $0 \leq r < C$ and, without loss of generality, we may assume that $g_1 \geq \cdots \geq g_n$ since the jobs share the same time window. Let $c_t = K_t - \sum_{j=(t-1)C+1}^{tC} g_j$ denote the cost of assigning the jobs $(t-1)C + 1; \cdots; tC$ to time slot t for $t \in \{1; \cdots; b\}$ and, if $r > 0$, let $c_{b+1} = K_{b+1} - \sum_{j=bC+1}^{n} g_j$ denote the cost of assigning the jobs $bC + 1; \cdots; n$ to time slot $b + 1$. This problem is solved by the following greedy algorithm that fills the next cheapest available time slot with the maximum number of the most profitable remaining jobs as long as that operation yields some positive profit (recall that the time-slots in a band are assumed to be ordered according to non-decreasing setup costs).

```
t := 1; j := 1;
while (t ≤ min{T; b + 1}) and (c_t < 0) do
   if t ≤ b then
      assign the jobs j; ···; j + C − 1 to time slot t
      j := j + C; t := t + 1;
   else assign the jobs j; ···; n to time slot t
endwhile.
```

The *restricted single-band problem* is defined as the variant of the single-band problem with the additional constraint that the first l positions of the band are unavailable. By convention no setup cost is due for the time slots

with at least one unavailable position. This variant is solved in the same way by first filling the first time slot containing at least one available position with as many jobs as possible from the most profitable jobs and then filling the next slots in the same way while there is still at least an available slot and an available job.
If $I \subseteq J$ is a subset of jobs whose time window contains the band B_u, we denote by $\Gamma(I; u; l)$ the optimal solution value of the instance of the restricted single-band problem defined by the jobs of I and the band B_u when the first l positions of B_u are assumed to be unavailable.

3.1 The dynamic programming algorithm

Let $I = (n; C; r; d; g; K)$ be an instance of $\Pi(1)$. We define below a subproblem $P(k; u; v; l)$ where:

- $k \in \{0; \cdots; n\}$,
- u and v are band indices such that $1 \leq u \leq v \leq q$,
- l is the number of unavailable positions in the band B_v,

The jobs of $P(k; u; v; l)$ are those jobs j from $\{1; \cdots; k\}$ such that $a_{u-1} \leq r_j < a_v$ (released in one of the bands $B_u; \cdots; B_v$). The release times are the same as in the initial problem but the new deadline of job j is $\min(d_j; a_v)$. Moreover, the positions $1; \cdots; l$ of band B_v are assumed to be unavailable and no set-up cost is due when jobs are assigned to time slots containing at least one unavailable position. We note that the initial problem corresponds to $P(n; 1; q; 0)$ and we let $F(k; u; v; l)$ be the minimum cost of a solution of $P(k; u; v; l)$. Figure 4 illustrates the definition of $P(k; u; v; l)$.
Finally, if k can be scheduled in B_v (i.e.,$v \in D(k)$), we define the slight variant $Q(k; u; v; l)$ of the subproblem $P(k; u; v; l)$ which is the same problem with the additional constraint that job k must be scheduled in the band B_v. $G(k; u; v; l)$ is the minimum cost of a solution of $Q(k; u; v; l)$.

The following property is a simple dominance property concerning the way jobs are assigned to the band B_v in an optimal solution of $P(k; u; v; l)$.

Lemma 2 *There is an optimal solution $(E; s)$ of $P(k; u; v; l)$ such that if d jobs are scheduled in the band B_v, these jobs are assigned to positions $l + 1; \cdots; l + d$.*

Proof. — Assume that $(E; s)$ is an optimal solution of $P(k; u; v; l)$ such that d jobs are scheduled in the band B_v and that the position $l + 1$ of band B_v

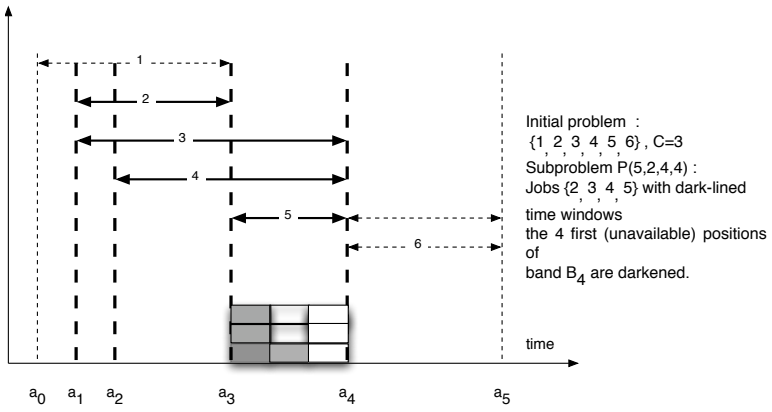

Figure 4: The subproblem $P(k;u;v;l)$

is free. Then one job j among the d jobs assigned to band B_v is assigned to position p such that $p > l+1$. By assigning j to position $l+1$ of B_v, we still get an optimal solution. This process may be repeated as long as one of the positions $l+1;\cdots;l+d$ is free. ∎

Note that Lemma 2 also applies to problem $Q(k;u;v;l)$. Moreover, the lemma implies that in an optimal solution, at most the first n time slots of a band may be assigned a job. So the number of used time slots is $O(n^2)$.

Let $f(k;v;l)$ be the cost of scheduling job k in the band B_v ($v \in D(k)$) when the first l positions of B_v are not available. We clearly have:

$$f(k;v;l) = \begin{cases} -g_k + K_t & \text{if } l \bmod C = 0 \\ -g_k & \text{otherwise} \end{cases}$$

where t is the first time slot of band B_v with at least one free position.
Let $(E;s)$ be a solution of problem $Q(k;u;v;l)$ such that job k is assigned position $l+1$ of band B_v and let be its cost. Let $'$ be the cost of the solution $(E \setminus \{k\};s')$ (where s' is the restriction of s to $E \setminus \{k\}$ of $P(k-1;u;v;l+1)$. The following property links and $'$.

Lemma 3 $= ' + f(k;v;l)$.

Proof. — Assume first that positions l and $l+1$ of B_v belong to the same time slot t. Then we have $= ' - g_k = ' + f(k;v;l)$. Conversely, il l and $l+1$ belong to consecutive time slots $t-1$ and t, the setup cost K_t is not taken

into account in the cost $\quad'$ of the solution $(E \setminus \{k\}; s')$ of $P(k-1; u; v; l+1)$ while it is part of the cost $\quad$ of the solution $(E; s)$ of $Q(k; u; v; l)$. We thus have $\quad = \quad' - g_k + K_t = \quad' + f(k; v; l)$. ∎

The following property establishes the link between the problems $P(k; u; v; l)$ and $Q(k; u; v; l)$ when $v \in D(k)$ (Recall that C_v is the number of positions in the band B_v).

Lemma 4 *If $v \in D(k)$ and $l < C_v$ then $G(k; u; v; l) = F(k-1; u; v; l+1) + f(k; v; l)$.*

Proof. — Let $(E; s)$ be an optimal solution of $Q(k; u; v; l)$ satisfying Lemma 2 and assume that job k is assigned to the position p of band B_v where $p > l+1$. By exchanging the jobs assigned to positions $l+1$ and p, we get an optimal solution $(E; s')$ of $Q(k; u; v; l)$ such that job k is assigned to position $l+1$ of B_v. So we can assume that in $(E; s)$ job k is assigned to the position $l+1$ of B_v. Consider now the solution $(E \setminus \{k\}; s')$ of problem $P(k-1; u; v; l+1)$ where s' is the restriction of s to $E \setminus \{k\}$ and assume that $(\hat{E};\ )$ is a better solution than $(E \setminus \{k\}; s')$. Then, from Lemma 3, $(\hat{E} \cup \{k\};\ ')$ where

$$'(j) = \begin{cases} (j) & \text{if } j \in \hat{E} \\ v & \text{if } j = k \end{cases}$$

and where job k is assigned to position $l+1$ of B_v, is a better solution of $Q(k; u; v; l)$ than $(E; s)$. So $(E \setminus \{k\}; s')$ is an optimal solution of $P(k-1; u; v; l+1)$. ∎

We now provide the main recurrence equation that is satisfied by $F(k; u; v; l)$. That equation is mainly derived from the property that if the band B_x to which job k is assigned is fixed, then the corresponding optimal solution is got by solving two subproblems each of which concerns the $k-1$ first jobs. So we denote by $F_x(k; u; v; l)$ the minimum cost value of a solution of $P(k; u; v; l)$ such that job k is assigned to the band B_x, with the convention that $x = 0$ means that job k is not selected. We thus get the following lemma.

Lemma 5 *Let $k \in \{1; \cdots; n\}$, $1 \leq u \leq v \leq q$ and $x \in D(k) \cap \{u; \cdots; v\} \cup \{0\}$.*

$$F_x(k; u; v; l) = \begin{cases} F(k-1; u; v; l) & \text{if } x = 0, \\ G(k; u; x; 0) + F(k-1; x+1; v; l) & \text{if } x \neq v, \\ F(k-1; u; v; l+1) + f(k; v; l) & \text{if } x = v. \end{cases}$$

Proof. — Assume $(E;s)$ is the best solution of $P(k;u;v;l)$ such that job k is not selected. Then clearly $(E;s)$ is an optimal solution of $P(k-1;u;v;l)$, which proves the first case of the lemma.
Assume now that $x = v$ and that $(E;s)$ is the best solution (satisfying Lemma 4) of $P(k;u;v;l)$ such that job k is scheduled in band B_v. In that case $(E;s)$ is an optimal solution of $Q(k;u;v;l)$. So, from Lemma 4, we get the third case of the lemma.
Assume now that $x \neq v$ and that $(E;s)$ is the best solution (satisfying Corollary 1) of $P(k;u;v;l)$ such that job k is scheduled in band B_x. Let us denote by E_1 the jobs of E whose release date is at most equal to a_{x-1}, let $E_2 = E \setminus E_1$ and notice that for all $j \in E_2$, we have $r_j \geq a_x$. Analogously let s_1 (respectively s_2) be the restriction of s to E_1 (respectively E_2).
From Lemma 1, we know that E_1 is a subset of the jobs of subproblem $Q(k;u;x;0)$. Assume that $(E'_1;\ \sigma_1)$ is a better solution of $Q(k;u;x;0)$ than $(E_1;s_1)$. Then $(E'_1 \cup E_2;\ \ )$ where

$$\sigma(j) = \begin{cases} \sigma_1(j) & \text{if } j \in E'_1 \\ s(j) & \text{if } j \in E_2 \end{cases}$$

is a better solution of $P(k;u;v;l)$ than $(E;s)$. Thus the cost of $(E_1;s_1)$ is equal to $G(k;u;x;0)$.
Again from Lemma 1, we know that E_2 is a subset of the jobs of subproblem $P(k-1;x+1;v;l)$. Assume that $(E'_2;\ \sigma_2)$ is a better solution of $P(k-1;x+1;v;l)$ than $(E_2;s_2)$. Then $(E_1 \cup E'_2;\ \ )$ where

$$\sigma(j) = \begin{cases} \sigma_2(j) & \text{if } j \in E'_2 \\ s(j) & \text{if } j \in E_1 \end{cases}$$

is a better solution of $P(k;u;v;l)$ than $(E;s)$. Thus the cost of $(E_2;s_2)$ is equal to $F(k-1;x+1;v;l)$.
So the cost of $(E;s)$, which is the sum of the costs of $(E_1;s_1)$ and $(E_2;s_2)$, equals $G(k;u;x;0) + F(k-1;x+1;v;l)$, what proves the second case of the lemma. ∎

We may now derive the recurrence equation satisfied by the function $F(k;u;v;l)$.

Theorem 2 *Let $k \in \{1;\cdots;n\}$ and $1 \leq u \leq v \leq n$.*
If $v \in D(k)$, then $F(k;u;v;l)$ is the minimum of the 3 values:

$$\begin{cases} F(k-1;u;v;l) \\ \min_{x \in D(k) \setminus \{v\}} \{F(k-1;u;x;1) + F(k-1;x+1;v;l) + f(k;x;0)\} \\ F(k-1;u;v;l+1) + f(k;v;l) \end{cases}$$

If $v \notin D(k)$*, then* $F(k;u;v;l)$ *is the minimum of the* 2 *values:*

$$\begin{cases} F(k-1;u;v;l) \\ \min_{x \in D(k) \cap \{u,\cdots,v\}} \{F(k-1;u;x;1) + F(k-1;x+1;v;l) + f(k;x;0)\} \end{cases}$$

Proof. — The proof basically relies on Lemma 5 since by partionning the solutions set of $P(k;u;v;l)$ according to the index value x of the band where job k is processed (or $x = 0$ if job k is rejected), we get :

$$F(k;u;v;l) = \begin{cases} \min_{x \in \{0\} \cup \{u,\cdots,v-1\} \cup \{v\}} F_x(k;u;v;l) & \text{if } v \in D(k) \\ \min_{x \in \{0\} \cup (\{u,\cdots,v\} \cap D(k))} F_x(k;u;v;l) & \text{if } v \notin D(k) \end{cases}$$

If $x = 0$ or $x = v$ then Lemma 5 directly gives the value of $F(k;u;v;l)$ else the value of $F(k;u;v;l)$ comes from Lemmas 5 and 4. Note finally that if $D(k) \setminus \{v\}$ or $D(k) \cap \{u;\cdots;v\}$ is the empty set, then the corresponding term is not taken into account in the minimum. ∎

Let us now provide the initial values of $F(k;u;v;l)$. We have $F(0;u;v;l) = 0$ since no job has to be processed. If $v > u$, then we have $F(k;u;v;C_v) = F(k;u;v-1;0)$ since no additional job may be scheduled in band B_v. Moreover, if $I(k;u)$ is the set of the jobs of $P(k;u;u;l)$, we have $F(k;u;u;l) = \Gamma(I(k;u);u;l)$ since in that case the problem reduces to the restricted single-band problem for the jobs of $I(k;u)$), the band B_u and the first l positions left unavailable.
The recurrence equation of Theorem 2 yields a dynamic programming algorithm with worst-case complexity $O(n^5)$ since $k \in \{1;\cdots;n\}$, $1 \leq u \leq v \leq q \leq 2n$, $Card(D(k)) \leq 2n$ and we may assume that $1 \leq l \leq C \leq n$ since no more than n positions of a band will be assigned.

3.2 Agreeable time windows

In this section, we consider the special case $\Pi(2)$ of $\Pi(1)$ when the time windows of the jobs are *agreeable* and we provide a more efficient dynamic programming algorithm for this case. The time windows are agreeable if:

$$r_i \leq r_j \Rightarrow d_i \leq d_j;$$

It is assumed that the jobs are indexed so that $r_1 \leq \cdots \leq r_n$ and we recall that $(a_0;\cdots;a_q)$ is the ordered list of the distinct release times and deadlines.

The algorithm to solve $\Pi(2)$ basically relies on the following dominance property, which is stronger than Lemma 1:

Lemma 6 *Let* $I = (n; C; r; d; g; K)$ *be an instance of* $\Pi(2)$. *There is an optimal solution* $(E; s)$ *such that* $i; j \in E$ *and* $i < j$ *implies* $s(i) \leq s(j)$.

Proof. — Assume that an optimal solution $(E; s)$ is such that for two jobs i and j in E we have $s(i) > s(j)$ and $i < j$. From the agreeable time windows assumption, we have $r_i \leq r_j < s(j) < s(i) \leq d_i \leq d_j$. By exchanging the time slots of the jobs i and j, we still get an optimal solution. We may then repeat the above transformation until the property is satisfied. ∎

Let us define for $j \in \{1; \cdots; n\}$, $t \in \{1; \cdots; q\}$ and $r_j < a_t \leq d_j$ the problem $P(j; t)$ as follows. The jobs of $P(j; t)$ are $1; \cdots; j$. If i is a job of $P(j; t)$, then its release time is r_i and its deadline is $\min\{d_i; a_t\}$. We note that $P(n; q)$ is the initial problem. Moreover we denote by $a(j; t)$ the first job of $P(j; t)$ whose deadline equals a_t, by $e(j; t)$ the number of jobs of $P(j; t)$ whose release time equals a_{t-1} and by $d(j; t)$ the largest deadline smaller than a_t among the jobs of $P(j; t)$. Figure 5 illustrates the definitions of $P(j; t)$, $a(j; t)$, $e(j; t)$ and $d(j; t)$.

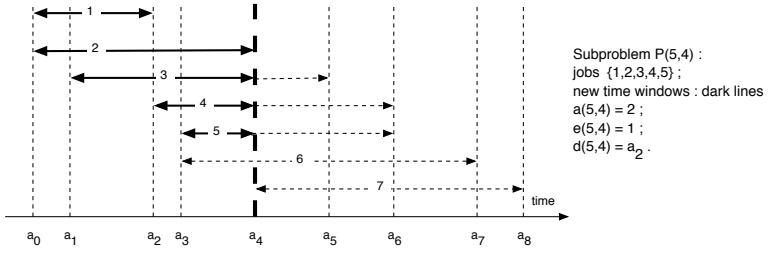

Figure 5: Subproblem $P(j; t)$

Let us denote the optimal solution value of $P(j; t)$ by $F(j; t)$. The recurrence equation whose proof is given in the appendix is as follows:

Theorem 3 *Let* $j \in \{1; \cdots; n\}$, $t \in \{1; \cdots; q\}$ *and* $r_j < a_t \leq d_j$.

$$F(j; t) = \min \begin{cases} F(j - e(j; t); t - 1) \\ \min_{a(j,t) \leq i \leq j} \{F(\ _{j,t}(i);\ _{j,t}(i)) + \Gamma(\{i + 1; \cdots; j\}; t; 1) + f(i; t; 0)\} \end{cases}$$

where

$$_{j,t}(i) = \begin{cases} t - 1 & \text{if } a(j; t) < i \leq j, \\ d(j; t) & \text{if } i = a(j; t) \end{cases}$$

$$_{j,t}(i) = \begin{cases} i - 1 & \text{if } a(j; t) \leq i \leq j - e(j; t) + 1, \\ j - e(j; t) & \text{if } j - e(j; t) + 1 < i \leq j \end{cases}$$

and $F(j;1) = \Gamma(\{1;\cdots;j\};1;0)$.

In order to improve the worst-case time complexity of the dynamic programming algorithm issued from Theorem 3, we may compute all the values $a(j;t)$, $e(j;t)$, $d(j;t)$ and all the values $\Gamma(\{i;\cdots;j\};u;1)$ as a preliminary step of the algorithm. Note first that for fixed j and t, the values $a(j;t)$, $e(j;t)$ and $d(j;t)$ can be computed in $O(n)$ time. So computing all these values takes $O(n^3)$ time. To compute the values $\Gamma(\{i;\cdots;j\};u;1)$, we first compute the cheapest n time slots of each band, which takes $O(n^2 \log n)$ time. Then we compute the list of the n jobs sorted by non-increasing gains in $O(n \log n)$. Now, given two jobs i and j such that $1 \leq i \leq j \leq n$, we compute the list $L(i;j)$ of the jobs $\{i;\cdots;j\}$ sorted by non-increasing gains in $O(n)$ time. Once $L(i;j)$ is known, computing the values $\Gamma(\{i;\cdots;j\};u;1); u \in \{1;\cdots;q\}$ takes $O(n^2)$ time. Thus computing all the values $\Gamma(\{i;\cdots;j\};u;1)$ for all $u \in \{1;\cdots;q\}$ takes $O(n^4)$ time. Since the complexity of computing the values $F(j;t)$ from Theorem 3 (once the values $a(j;t)$, $e(j;t)$, $d(j;t)$ and $\Gamma(\{i;\cdots;j\};u;1); u \in \{1;\cdots;q\}$ are known) takes $O(n^3)$ time, the overall complexity of the algorithm is $O(n^4)$ time.

3.3 Agreeable time windows and equal gains

We now consider the special case $\Pi(3)$ of $\Pi(2)$ where all the jobs have the same gain $\bar{g}$. Let an *interval of jobs* be a set of jobs with consecutive indices. The dominance property of Lemma 6 may now be further reinforced as follows:

Lemma 7 *Let* $I = (n;C;r;d;\bar{g};K)$ *be an instance of* $\Pi(3)$. *There is an optimal solution* $(E;s)$ *such that the jobs performed in each band make an interval of jobs.*

Proof. — Let $(E;s)$ be an optimal solution satisfying Lemma 6 and assume that that there is a band B_t such that the jobs performed in B_t do not make an interval. If f (respectively l) is the first(respectively last) job performed in B_t, there is a job k ($f < k < l$) such that k is not processed in B_t, which implies from Lemma 6 that $k \notin E$. Moreover by the agreeable structure of the time windows we have $t \in D(k)$ since $t \in D(f); t \in D(l)$ and $f < k < l$. Thus $(E \setminus \{f\} \cup \{k\}; s')$ where:

$$s'(i) = \begin{cases} s(i) & \text{if } i \in E \setminus \{f\} \\ s(f) & \text{if } i = k \end{cases}$$

is a feasible solution of I. Since i and k have the same cost and since the number of jobs assigned in each band is the same in $(E;s)$ as in $(E \setminus \{f\} \cup \{k\}; s')$,

$(E \setminus \{f\} \cup \{k\}; s')$ is also an optimal solution. However the cumulated length of the holes in the index set of $E \setminus \{f\} \cup \{k\}$ has been decreased by one unit. So, we may repeat the above transformation until an interval of jobs is performed in each band. ∎

From Lemma 7, we know that the set of jobs processed in each band B_t is an interval of jobs $f; f+1; \cdots; l$. These $l - f + 1$ jobs are clearly scheduled in the first $\lceil (l - f + 1)/C \rceil$ time slots of band B_t.
The common-gain assumption yields a further simple remark concerning the structure of of an optimal solution of the problem $P(j;t)$. If the interval of the jobs scheduled in the band B_t contains d jobs ($d \geq 1$), then we may assume that these jobs are the last ones of $P(j;t)$,i.e: $j - d + 1; \cdots; j$.
Let $B_t^j = \{i | i \leq j; t \in D(i)\}$ be the set of jobs preceeding job j that may be executed in band B_t, let $b_t^j = Card(B_t^j)$ and recall that C_t is the number of positions in band B_t. If $d \in \{1; \cdots; C_t\}$, we denote by $\Gamma(t;d)$ the cost of scheduling d jobs in band B_t. Then the optimal solution value $f(j;t)$ of $P(j;t)$ satisfies the following recurrence equation whose proof is omitted since quite similar to that of Theorem 3.

Theorem 4 *Let* $j \in \{1; \cdots; n\}$, $t \in \{1; \cdots; q\}$ *and* $a_t > r_j$.

$$f(j;t) = \min \begin{cases} f(j-1;t); \\ \min_{d \in \{0, \cdots, \min\{b_t^j, C_t\}\}} \{f(j-d; t-1) + \Gamma(t;d)\} \end{cases}$$

where for any $t \in \{1; \cdots; q\}$, $f(0;t) = 0$.

As a first step of the algorithm issued from Theorem 4, we first compute the n cheapest time slots of each band in $O(n^2 \log n)$ time and then compute all the values $\Gamma(t;d)$ in $O(n^2)$ time. The overall complexity of the algoritm is then $O(n^3)$.

4 Conclusion

In this paper, we have studied the combination of rejection, time windows and capacity constraints to schedule a set of unit length jobs with the additional feature that a non-empty time slot incurs a specific setup cost and a selected job induces a specific gain. The case of time-dependent job gains, where the main dominance relation is no longer satisfied, should be an interesting problem. Further research will also deal with other compatibility constraints such as compatibility graphs.

Acknowledgements

We thank the anonymous referees for their helpful remarks and comments.

References

[1] P. Baptiste and C. Le Pape, Scheduling a single machine to minimize a regular objective function under setup constraints, *Discrete Optimization* **2**, 83–99.

[2] T. Benoist, E. Bourreau, Y. Caseau and B. Rottembourg (2001). Towards Stochastic Constraint Programming: A Study of On-Line Multi-Choice Knapsack with Deadlines. *Proceedings CP'01, LNCS 2239*, 61–76, Springer 2001.

[3] P. Baptiste, P. Brucker, S. Knust and V. Timkovsky(2004), Ten notes on equal-processing-time scheduling, *4OR Quarterly Journal of the Belgian, French and Italian Operations Research Societies* **2**, 111–127.

[4] M. van den Akker and H. Hoogeveen (2004), Minimizing the number of tardy jobs, in J.Y-T. Leung (ed), *Handbook of Scheduling*, chapter 12.

[5] D.W. Engels, D.R. Karger, S.G. Kolliopoulos, S. Sengupta, R.N. Uma and J. Wein (2003), Techniques for scheduling with rejection, *Journal of Algorithms* **49**, 175 – 191.

[6] Y. Bartal, S. Leonardi, A. Marchetti-Spaccamela,J. Squal and L. Stougie (2003), Multiprocessor scheduling with rejection, *SIAM Journal on Discrete Mathematics* **13**,1, 64 – 78.

[7] S. Seiden (2001), Preemptive multiprocessor with rejection, *Theoretical Computer Science* **262**,1 437 – 458.

[8] D. György and H. Yong(2006), Scheduling with machine cost and rejection, *Journal of Combinatorial Optimization* **12**,4 337 – 350.

[9] F. Gardi (2008), Mutual exclusion scheduling with interval graphs or related classes, *Discrete Applied Mathematics* **to appear**.

[10] G. Finke, V. Jost, M. Queyranne and A. Sebö (2004), Batch processing with interval graphs compatibilities between tasks, *Cahiers du laboratoire Leibniz* **108**, to appear in Disrete Applied Mathematics.

[11] H. Hoogeven, M. Skutella and G.J. Woeginger (2003), Preemptive scheduling with rejection, *Mathematical Programming* **94**,2-3 361 – 374.

Appendix: Proof of Theorem 3

Proof. — Let $(E;s)$ be the best solution of $P(j;t)$ such that no job of $P(j;t)$ is scheduled in band B_t. None of the $e(j;t)$ last jobs of $P(j;t)$ is in E. So $(E;s)$ is also an optimal solution of $P(j-e(j;t);t-1)$.
Let now $(E;s)$ be the best solution of $P(j;t)$ such that the first job of $P(j;t)$ scheduled in band B_t is job i where $a(j;t) \leq i \leq j$. By Lemma 6, any job k where $i+1 \leq k \leq j$ is either not selected or scheduled in band B_t. So the restriction of $(E;s)$ to $\{i+1;\cdots;j\}$ is an optimal solution of the single-band problem with jobs $\{i+1;\cdots;j\}$, band B_t and one unavailable position. The cost induced by the jobs $\{i;\cdots;j\}$ in $(E;s)$ is thus $\Gamma(\{i+1;\cdots;j\};t;1)+f(i;t;0)$.
Again by Lemma 6, we know that any job k where $1 \leq k < i$ is either not selected or scheduled in a band B_u with $u < t$. More precisely if $a(j;t) < i \leq j-e(j;t)+1$, then we know that $i-1$ is a job of $P(i-1;t-1)$. If $i = a(j;t)$ then $i-1$ is a job of $P(i-1;d(j;t))$. If $j-e(j;t)+1 < i \leq j$, then the jobs $j-e(j;t)+1;\cdots;i-1$ are not in E. From the definition of $_{j,t}(i)$ and $_{j,t}(i)$, we have that the restriction of $(E;s)$ to $\{1;\cdots;\ _{j,t}(i)\}$ is an optimal solution of $P(\ _{j,t}(i);\ _{j,t}(i))$. So the cost of $(E;s)$ is $F(\ _{j,t}(i);\ _{j,t}(i))+\Gamma(\{i+1;\cdots;j\};t;1)+f(i;t;0)$.
The cost of the best solution of $P(j;t)$ such that at least one job is scheduled in band B_t is thus $\min_{a(j,t)\leq i\leq j}\{F(\ _{j,t}(i);\ _{j,t}(i))+\Gamma(\{i+1;\cdots;j\};t;1)+f(i;t;0)\}$. ∎