

# Selecting and Scheduling Tasks with Agreeable Time Windows and Setup Costs

Philippe Chrétienne, Francis Sourd

Universit Paris 6, LIP6, France, {Philippe.Chretienne, Francis.Sourd}@lip6.fr

## 1 Introduction

The problem studied in this paper originates in a satellite launcher problem. When a satellite launch is ordered to a launching company, three important parameters have to be taken into account : the time window during the satellite must be launched, the satellite mass and the cost of the satellite launch. When a given set of launches has to be planned, the company has to decide which satellites will be launched and when these selected launches will be scheduled, with the objective of maximizing its profit. When a non empty set of satellite launches is assigned to a time slot, the company has to pay for the installation of a launching platform. Moreover, the total mass of the launched satellites must not exceed the launching capacity of the platform.

In a formal description of the problem, we consider  $n$  unit-length independent jobs  $J_1, \dots, J_n$  each of which represents a satellite and has an individual weight  $w_i$  (to be understood as the weight of an item in a knapsack problem). For each job  $J_i$ , it must be decided whether it will be processed or not and, in the positive case, in which time-slot of an a priori given time window  $[r_i, d_i]$  it must be performed. A positive *gain*  $g_i$  results from deciding to perform  $J_i$  and a positive fixed *setup* cost  $K_t$  is due if time slot  $[t - 1, t]$  (also denoted by  $t$ ) is assigned to at least one job. Finally, the sum of the weights of the jobs performed in any time-slot must not exceed a given capacity  $C$ . A solution  $S$  of the problem is a pair  $(E, s)$  where :

- $E$  is the subset of the executed jobs,
- $s : E \mapsto T$  indicates the time-slots assigned to the jobs of  $E$  (to simplify the notation, we shall write  $s(i)$  for  $s(J_i)$ ),
- for any  $J_i \in E$ ,  $s(i)$  must be in  $[r_i + 1, d_i]$
- for any time-slot  $t$  in  $s(E)$ , the sum  $\sum_{\{j|s(j)=t\}} w_j$  must be less than  $C$ .

The cost of the solution  $(E, s)$  is  $\sum_{t \in s(E)} K_t - \sum_{j \in E} g_j$  (the opposite of this expression is of course the total benefit) and the problem is to find a minimum-cost solution.

Our problem is clearly NP-complete since finding if there is a schedule with makespan less than 2 such that all the jobs are processed can be reduced to the PARTITION problem. We can also reduce 3-PARTITION to our problem, which proves that it is strongly NP-hard. The goal of this work is to study some polynomial special cases. Some cases are special cases of classical scheduling problems.

- The single-machine case, that is  $C = 1$  and  $w_j = 1$  for all  $j$  is easy. As at most one job can be scheduled in any time slot, we simply have an assignment problem between jobs and time-slots where the cost to assign slot  $t$  to job  $J_j$  is given by  $K_t - g_j$ .
- When there are no setup costs, that is  $K_t = 0$  for all  $t$ , the objective function is equivalent to the sum of tardy jobs since a tardy job can be regarded as a non-processed job. Van den

Akker and Hoogeveen propose a survey of these problems [2]. Other authors have also studied the presence of rejection penalties in addition to other scheduling costs such as the weighted flow time [3] or the sum of tardiness [1].

In this paper we propose a dynamic-programming polynomial algorithm for the special case with unit weights, a common gain (i.e:  $g_j = g$ ) and agreeable time windows. We recall that time windows are agreeable if  $r_i < r_j \Rightarrow d_i \leq d_j$ . It means that jobs are numbered in the order of the release dates (and deadlines).

## 2 The dynamic programming algorithm

We first propose a simple dominance property that is valid even without the assumption of agreeable time windows.

**Property 1** *There is an optimal solution  $(E, s)$  such that for any two jobs  $J_i$  and  $J_j$  in  $E$  such that  $s(i) < s(j)$ , then either  $d_i \leq d_j$  or  $s(i) + 1 \leq r_j$ .*

*Proof.* — Assume that an optimal solution  $(E, s)$  is such that for two jobs  $J_i$  and  $J_j$  in  $E$  we have  $s(i) < s(j)$ ,  $d_i > d_j$  and  $s(i) + 1 > r_j$ . By exchanging the time slots of the jobs  $J_i$  and  $J_j$ , we still get an optimal solution. We may then repeat the above transformation until the property is satisfied. ■

The next dominance property basically relies on the dominance property:

**Property 2** *There is an optimal solution  $(E, s)$  such that  $i < j \Rightarrow s(i) \leq s(j)$ .*

*Proof.* — Assume that an optimal solution  $(E, s)$  is such that for two jobs  $J_i$  and  $J_j$  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  $J_i$  and  $J_j$ , we still get an optimal solution. We may then repeat the above transformation until the property is satisfied. ■

Let  $(a_0, \dots, a_q)$  be the ordered list of the distinct release times and deadlines. Clearly,  $q < 2n$ .

**Property 3** *There is an optimal solution  $(E, s)$  such that if the jobs  $J_i$  and  $J_j$  are scheduled in the time interval  $[a_{t-1}, a_t]$ , then any job  $J_k$  with  $i < k < j$  is also scheduled in this interval.*

*Proof.* — Consider an optimal schedule satisfying Property 2. Assume that  $J_k$  is not processed in  $[a_{t-1}, a_t]$  while  $J_i$  and  $J_j$  ( $i < k < j$ ) are scheduled in this interval. From Property 2, we know that  $J_k$  is not processed. Since there is no deadline in the interval,  $J_k$  is available at any time point. Therefore we can replace  $J_j$  by  $J_k$  and eventually reorder the jobs according to their index. We may then repeat the above transformation until the property is satisfied. ■

As a consequence of this property, the set of jobs processed in any  $[a_{t-1}, a_t]$  is an interval of jobs  $J_i, J_{i+1}, \dots, J_j$ . These  $j - i + 1$  jobs are clearly scheduled in  $n_{ij} = \lceil (j - i + 1)/C \rceil$  time-slots which correspond to the smallest  $n_{ij}$  values of the setup costs in this interval. Therefore, the associated cost only depends on the number of jobs in the interval and is accordingly denoted by  $\kappa_t(j - i + 1)$ . By convention, if  $j - i + 1 > (a_t - a_{t-1})C$ , then we cannot schedule all these jobs and we set  $\kappa_t(j - i + 1) = +\infty$ .

For any  $t \in \{0, \dots, q\}$  and any  $j \in \{0, \dots, n\}$ , we define the subproblem  $\Pi(j, t)$  as follows: the jobs of  $\Pi(j, t)$  are  $\{J_1, \dots, J_j\}$  and for each job  $J_i$ , its release time is still  $r_i$  and its deadline becomes  $\min\{a_t, d_i\}$ . Note that  $\Pi(n, q)$  is the original problem.

Along with subproblem  $\Pi(j, t)$ , we define  $\pi(j, t)$  to be the value of an optimal solution of  $\Pi(j, t)$  and we show that  $\pi(j, t)$  satisfies a recurrence equation.

Let us first observe that an optimal solution of  $\Pi(j, t)$  satisfies Property 3. Therefore, in such an optimal solution, the jobs scheduled in  $[a_{t-1}, a_t]$  are  $J_i, \dots, J_{i'}$ . If  $d_j < a_t$ , then  $d_j \leq a_{t-1}$  and no job among  $\{J_1, \dots, J_j\}$  can be scheduled in the last interval  $[a_{t-1}, a_t]$ . Therefore  $\pi(j, t) = \pi(j, t-1)$ .

Let us now assume that  $d_j \geq a_t$ . If  $i' \neq j$ , we can get a new optimal schedule by replacing the last  $i' - i + 1$  jobs by the jobs  $J_{i-i'+j}, \dots, J_j$ . Therefore, if there are  $l$  jobs in the last interval  $[a_{t-1}, a_t]$ , then the cost of the schedule is given by  $\pi(j-l, t-1) + \kappa_t(l)$ . Since all the jobs in  $[a_{t-1}, a_t]$  must be release before  $a_{t-1}$ , we have:

$$\pi(j, t) = \begin{cases} \pi(j, t-1) & \text{if } d_j < a_t \\ \min\{\pi(j-l, t-1) + \kappa_t(l) \mid 0 \leq l \leq j \text{ and } r_{j-l+1} \geq a_{t-1}\} & \text{if } d_j \geq a_t \end{cases}$$

We clearly have  $\pi(j, 0) = 0$  for all  $j$  and  $\pi(0, t) = 0$  for all  $t$ . From these values, the dynamic program can be run.

All the functions  $\kappa_t$  are piecewise constant and can be built in  $O(T \log T)$  time where  $T = a_q$  is the horizon of the schedule. Each value  $\pi(j, t)$  can be computed in  $O(n)$  time, which means that all the values are computed in  $O(n^3)$  time. Therefore, the time complexity of the algorithm is  $O(T \log T + n^3)$ . In the special case where all the setup costs are equal to a common constant  $K$ , the values  $\kappa_t(l)$  can be evaluated in constant time and the time complexity is in  $O(n^3)$ .

## References

- [1] Philippe Baptiste and Claude Le Pape, Scheduling a single machine to minimize a regular objective function under setup constraints, *Discrete Optimization* **2**, 83 – 99.
- [2] 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.
- [3] 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.