Scheduling problems with processing time dependent profit: applications and a nice polynomial case

Florian Fontan, Nadia Brauner, Pierre Lemaire

Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, F-38000 Grenoble, France
{firstname.lastname}@grenoble-inp.fr

Keywords: scheduling, controllable processing time, polynomial algorithm, b-matching

We study the complexity of scheduling problems where jobs have a variable processing time: one can *decide* the processing time of each job. The profit for a job then depends on its allocated processing time. Detailed results and proofs for this problem can be found in the thesis of Fontan (2019). This abstract presents an insight of that document.

In our experience, the problem originates from astrophysics and the search for exoplanets (Lagrange *et. al.* 2016). Astrophysicists want to schedule observations on a telescope and, for each possible target (star), there exist time-windows when it is visible, a required duration for its observation, and an interest for observing it; the objective is to maximize the total interest of the schedule. This primary version of the problem has been described and solved by Catusse *et. al.* (2016); but it appears that shortening an observation would be worth doing if that makes room for another one. More generally an observation remains relevant even if its processing (observation) time is slightly less than the required value, with an accordingly downgraded interest. Such a situation can be modeled by processing time dependent profits. The general problem is NP-complete, and an efficient practical solution algorithm has been proposed Fontan (2019). In the following, we present the model for processing time dependent profit (Section 1), limited to the case where time-windows only differ by their deadlines. Then in Section 2, we focus on a special polynomial case to show a proof technique for those problems.

1 Processing time dependent profit

We consider a scheduling problem with n jobs and m identical parallel machines; each job T_j has a deadline d_j and a profit function $w_j(p_j)$ that depends on the decided processing time p_j. Figure 1 shows three examples of profit functions.

A schedule is feasible if it satisfies the following conditions:

- every machine processes only one job at a time,
- a scheduled job T_j must start after 0 and end before its deadline d_j,
- preemption is not allowed.

The objective is to find a feasible schedule that maximizes the total profit:

$$\max \sum_{j=1}^{n} w_j(p_j)$$

with the convention that $p_j = 0$ if T_j is not scheduled.

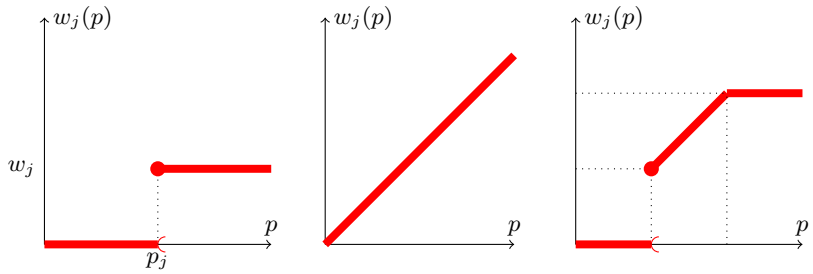
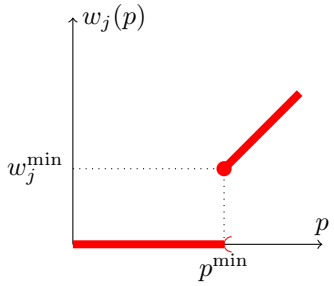


Fig. 1. Examples of profit functions of: (a) a classical scheduling problem ($w_j(p) = w_j$ if $p \geq p_j$, 0 otherwise); (b) a basic (linear profit) problem; (c) the star observation problem

2 Focus on a polynomial case solved with maximum weight b-matching

In this section, we focus on a specific variant with a common deadline, *i.e.* d_j=d for all jobs T_j, and the following profit function:

$$w_j(p) = \begin{cases} 0 & \text{if } p < p^{\min} \\ w_j^{\min} + b_j(p - p^{\min}) & \text{if } p \geq p^{\min} \end{cases}$$



We exhibit a polynomial algorithm for this special processing time dependent profit maximization scheduling problem with parallel machines. This algorithm uses as subproblem the maximum weight b-matching which can be solved in polynomial time (Schrijver 2002): Given a graph $G(V, E)$, a demand/supply b_v for each vertex $v \in V$, and a weight/cost c_e for each edge $e \in E$, a b-matching of G is a vector $x \in \mathbb{N}^E$ such that $\sum_{u,(uv)\in E} x_{(uv)} \leq b_v$ for all $v \in V$. The weight of a b-matching $x \in \mathbb{N}^E$ is defined as $\sum_{e\in E} c_e x_e$. The maximum weight b-matching problem is then the problem of finding a b-matching of maximum weight in G.

A set of solutions is dominant if it contains at least one optimal solution. Our algorithm consists in solving a polynomial number of maximum weight b-matching problems which rely on the dominant set described below.

We consider the case $d = qp^{\min} + r$, with $q, r \in \mathbb{N}$, $q \geq 2$, $0 < r < p^{\min}$. However, with a similar reasoning, the results can be adapted for the case $d = qp^{\min}$, $q \in \mathbb{N}$, $q \geq 2$. The case $d < 2p^{\min}$ is trivial.

Lemma 1. *Let $\mathcal{U}$ be the set of solutions such that for all $S \in \mathcal{U}$:*

- *for all $T_j \in S$, $p_j \geq p^{\min}$;*
- *on each machine, there exists at most one job $T_j \in S$ such that $p_j \neq p^{\min}$;*
- *there exists at most one job $T_j \in S$ such that $p_j \notin \{d, p^{\min}, p^{\min} + r\}$. Such a job is called a* special *job.*

Then, $\mathcal{U}$ is dominant.

Figure 2 illustrates the structure of the solutions of $\mathcal{U}$. The horizontal axis corresponds to the time, each line represents a machine and each striped rectangle and its length

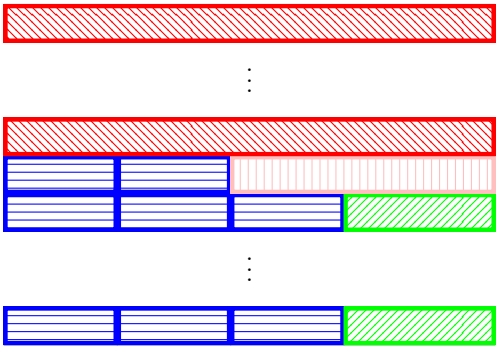


Fig. 2. Illustration of the structure of the solutions in $\mathcal{U}$

represent a scheduled job and its processing time. The long jobs on the top machines are those of length d; the smallest jobs are those of length $p^{\min}$; the jobs at the end of the bottom machines are those of length $p^{\min}+r$; the job with vertical stripes is a special job.

Let $S \notin \mathcal{U}$ be an optimal solution. The idea of the proof of Lemma 1, that we do not detail here, is to build another solution $S' \in \mathcal{U}$ from S without degrading its value.

Theorem 1. *The processing time dependent profit scheduling problem with a common deadline for all jobs and a profit function as decribed at the beginning of this section can be solved in polynomial time.*

Proof. Following Lemma 1, we only focus on solutions of $\mathcal{U}$. Thus, we can partition the jobs of a solution $S \in \mathcal{U}$ into 4 subsets depending on their processing time (and one more for the non scheduled jobs):

$$\begin{aligned}
U_1(S) &= \{T_j \in S, \quad p_j = d\} \\
U_2(S) &= \{T_j \in S, \quad T_j \text{ is a special job}\} \\
U_3(S) &= \left\{T_j \in S, \quad p_j = p^{\min} + r\right\} \\
U_4(S) &= \left\{T_j \in S, \quad p_j = p^{\min}\right\} \\
U_0(S) &= \{T_j \notin S\}
\end{aligned}$$

In addition, we define $n_k(S)$ the cardinality of $U_k(S)$ and $t(S)$ the number of jobs scheduled on the same machine as the job of U_2:

$$\forall k \in \{0, \ldots, 4\}, \quad n_k(S) = |U_k(S)|$$

$$t(S) = \begin{cases} 0, & \text{if } U_2(S) = \emptyset \\ \frac{d-p_j}{p^{\min}}, & \text{if } U_2(S) = \{T_j\} \end{cases}$$

Note that, if $U_2(S) = \{T_j\}$, then $p_j = d - t(S)p^{\min}$.

We now infer the following relations. For all $S \in \mathcal{U}$:

$$0 \leq n_1(S) \leq m \qquad 0 \leq t(S) \leq n-1$$

$$\begin{aligned}
n_2(S) &= \begin{cases} 0, \text{ if } t(S) = 0 \\ 1, \text{ otherwise} \end{cases} \\
n_3(S) &= m - n_1(S) - n_2(S) \\
n_4(S) &= (q-1)n_3(S) + t(S) \qquad \text{(remember that } q = \lfloor d/p^{\min} \rfloor) \\
n_0(S) &= n - n_1(S) - n_2(S) - n_3(S) - n_4(S)
\end{aligned}$$

Therefore, if $n_1(S)$ and $t(S)$ are fixed, we can determine $n_2(S)$, $n_3(S)$, $n_4(S)$ and $n_0(S)$. Thus, the optimal value is the best optimal value of the $O(n^2)$ problems with those parameters fixed:

$$\mathrm{OPT} = \max_{(n_1,t)\in(\{0,\ldots,m\}\times\{0,\ldots,n-1\})} \mathrm{OPT}(n_1,t)$$

Now, we show that for all $(n_1,t) \in \{1,\ldots,m\} \times \{0,\ldots,n-1\}$, a maximum weight b-matching model can compute $\mathrm{OPT}(n_1,t)$. We create n nodes T_j, $j = 1,\ldots,n$ with supply 1, and 5 nodes U_i, $i = 0,\ldots,4$ with demand n_i such that $n_2 = 0$ if $t \leq 1$, 1 otherwise; $n_3 = m - n_1 - n_2$, $n_4 = (q-1)n_3 + t$, $n_0 = n - n_1 - n_2 - n_3 - n_4$. Then for all $j = 1,\ldots,n$, for all $i = 0,\ldots,4$, we add the arc (T_j, U_i) and cost:

$$c_{ji} = \begin{cases} w_j(d) & i = 1 \\ w_j(d - tp^{\min}) & i = 2 \\ w_j(p^{\min} + r) & i = 3 \\ w_j^{\min} & i = 4 \\ 0 & i = 0 \end{cases}$$

A part of the graph including only one job is represented in Figure 3. Edges are only between a node corresponding to a job and a node corresponding to a set U_k. If edge (T_j, U_k) is used in the b-matching solution, then job T_j will be scheduled according to the corresponding set in the corresponding solution of the scheduling problem. For example, if $U_k = U_1$, then $p_j(S) = d$. Hence, the obtained solution thus corresponds to a schedule S with $n_1(S) = n_1$, etc.

The size of the graph is polynomial compared to the size of the instance. Furthermore, the number of problems that we have to solve is $O(n^2)$. Therefore, the problem can be solved in polynomial time.

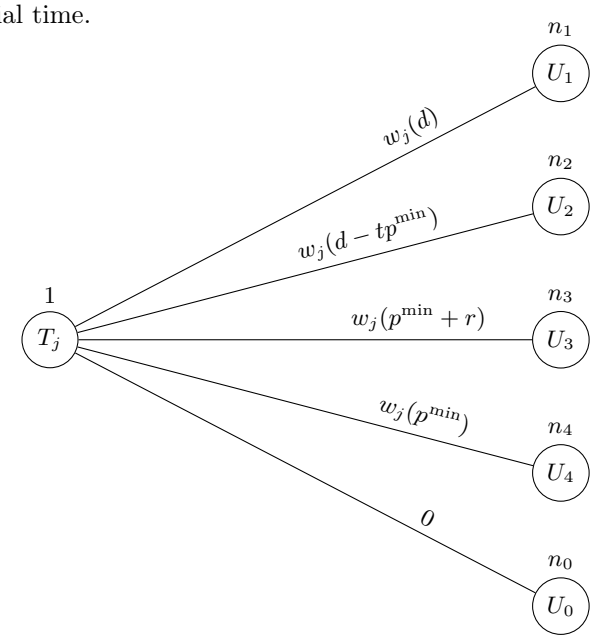


Fig. 3. A part of the graph showing only one job given as input of the maximum weight b-matching problem

References

Anne-Marie Lagrange, Pascal Rubini, Nadia Brauner, Hadrien Cambazard, Nicolas Catusse, Pierre Lemaire, and Laurence Baude. SPOT: an optimization software for dynamic observation programming. In *SPIE 9910, Observatory Operations: Strategies, Processes, and Systems VI, 991033 (July 18, 2016)*, Edinburgh, United Kingdom, July 2016.

Nicolas Catusse, Hadrien Cambazard, Nadia Brauner, Pierre Lemaire, Bernard Penz, Anne-Marie Lagrange, and Pascal Rubini. A Branch-And-Price Algorithm for Scheduling Observations on a Telescope. In *Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 3060–3066. AAAI Press, 2016.

Florian Fontan. *Theoretical and practical contributions to star observation scheduling problems.* PhD Thesis, Grenoble 2019.

Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2002.