Minimizing the Earliness and Tardiness Cost of a Sequence of Tasks on a Single Machine.

Philippe Chrétienne

March 23, 1999

Abstract

Assume that n tasks must be processed by one machine in a fixed sequence. The processing time, the preferred starting time and the earliness and tardiness costs per time unit are known for each task. The problem is to allocate each task a starting time such that the total cost incurred by the early and tardy tasks is minimum. Garey et al. have proposed a nice $O(n \log n)$ algorithm for the special case of symmetric and task-independent costs. In this paper we first extend that algorithm to the case of asymmetric and task-independent cost without increasing its worst-case complexity. For the general case of asymmetric and task-dependent costs, we propose an $O(n^3 \log n)$ algorithm based on a strong dominance property that yields to efficiently model the scheduling problem as a minimum cost path in a valued directed acyclic gaph.

1 Introduction

Due to their numerous applications, scheduling problems where the tasks incurred a cost both if they are early or tardy have received much attention. As an example, in a just-in-time production, a piece that is finished before its delivery time incurs an inventory cost while it incurs a backlog cost if it is finished after its delivery time. Moreover, there are many production systems where there is a priori no evidence for the inventory and backlog per time unit costs to be equal or not to depend on the individual tasks. Many variants of that problem have been studied [5],[1],[6],[7], [8],[9] and quite good surveys such as [2],[3],[4] show the amount and the diversity of the research in this field.

In this paper, we revisit the basic problem where a finite set of tasks must be processed on a single machine in a given order. Each task has a given preferred starting time and its earliness or tardiness in a schedule is the deviation about that preferred starting time. We assume that an early or tardy task incurs a cost which is proportional to the corresponding earliness or tardiness value. However, the corresponding per-time-unit earliness and tardiness costs need neither be equal nor be independent of the individual tasks.

The main reference for this problem concerns the secial case of symmetric and task-independent costs: Garey et al. [1] have developped a nice $(O(n \log n)$ algorithm that iterates a transformation that allows to compute an optimal schedule for the problem restricted to its $q+1$ first tasks from the problem restricted its first q tasks.

We propose here an algorithm with the same complexity that extends the algorithm in [1] to asymmetric costs. For the general problem with asymmetric and task-dependent costs , we use a convexity property of the cost function of an allocated block and a strong necessary condition on the starting times of the allocated blocks in an *strongly left-adjusted* optimal schedule to first model the problem as the search of a minimum-cost path in a directed acyclic graph called the *indivisible blocks* graph and then derive an $O(n^3 \log n)$ algorithm.

Section 1 defines the scheduling problem and its main notations. Section 2 briefly recalls the algorithm in [1] for symmetric and task-independent costs. Section 3 presents the extension of that algorithm to asymmetric and task-independent costs. Section 5 gives an algorithm for asymmetric and task-dependent costs.

2 Definitions and notations

n non-preemptive tasks $T_1, \cdots, T_n$ must be processed by a single machine in a given order, for example the order $(1, \cdots, n)$. For each task T_i, we denote by p_i its processing time, by ω_i its preferred starting time, and respectively by a_i and r_i its per time-unit earliness and tardiness costs. The task T_i started at time t_i incurs a cost $c_i(t_i)$ defined by:

$$c_i(t_i) = \begin{cases} a_i(\omega_i - t_i) & \text{if } t_i \leq \omega_i \\ r_i(t_i - \omega_i) & \text{if } t_i \geq \omega_i \end{cases}$$

The problem is to allocate a starting time to each task so as to minimize the total cost $\sum_{i=1}^{n} c_i(t_i)$.

A block of S is a left and right maximal list $B = (T_i, T_{i+1}, \cdots, T_j)$ of tasks performed without any intermediate delay in S. Thus a schedule S is also a list $((B_1, s_1), \cdots, (B_b, s_b))$ of (block,date) pairs called *allocated blocks* such that:

1. $B_1 \cdots B_b = (T_1, \cdots, T_n)$
2. for any $k \in \{2, \cdots, b\}$, $s_k > s_{k-1} + p(B_{k-1})$.

where $p(B_k) = \sum_{T_i \in B_k} p_i$ and where s_k is the starting time of the first task of B_k.
Let $S = ((B_1, s_1), \cdots, (B_b, s_b))$ be a schedule. We denote respectively by $b(S)$, $s_k(S)$, $f_k(S)$ and $n_k(S)$ the number of blocks, the starting time of the k^{th} block, the completion time of the k^{th} block and the index of the last task of the k^{th} block. When there is no ambiguity to which schedule they refer, the reference to S will be omitted in these notations.
If (B, s) is an allocated block, the subsets of the early tasks, on-time tasks and tardy tasks in (B, s) are respectively denoted by $\mathcal{A}(B, s)$, $\mathcal{H}(B, s)$ and $\mathcal{R}(B, s)$.
The cost of (B, s) is denoted by $c_B(s)$ whereas the cost of S is denoted by $c(S)$. The following property concerns the shape of the time function $c_B(t)$.

Property 1 *$c_B(t)$ is a convex and piecewise linear time function.*

Proof. — Let $B = (T_1, \cdots, T_K)$ and for any $k \in \{1, \cdots, K\}$ let π_k be eequal to $\sum_{i=1}^{k-1} p_i$ with by convention $\pi_1 = 0$. For any $k \in \{1, \cdots, K\}$, the starting time of task T_k within the allocated block (B, t) is thus $\pi_k + t$. Let X be a subset of the tasks in B, we denote by $r(X)$ the value $\sum_{T_i \in X} r_i$ and we define $h(X)$ and $a(X)$ in the same way. Let $A(t), H(t), R(t)$ be respectively the subsets of early, on-time and tardy tasks in (B, t).
Since the individual cost of the task T_i within (B, t) is a continuous time function on $[0, +\infty[$ (see Figure 1), $c_B(t)$ is also a continuous time function on $[0, +\infty[$.
Let $\theta_0 = 0$, we define θ_k as *the smallest* time $t > \theta_{k-1}$ such that at least one task in $A(\theta_{k-1})$ is on-time at t. Let A_k, H_k, R_k be the subsets of the early, on-time and tardy tasks in (B, θ_k) and let $c_k = c_B(\theta_k)$. We have for any $t \in [\theta_{k-1}, \theta_k[$:

$$c_B(t) = c_{k-1} + (t - \theta_{k-1})(r(R_0 \cup (\cup_{j=0}^{k-1} H_j) - a(A_0 \setminus \cup_{j=1}^{k-1} H_j)).$$

Let $u_k = (r(R_0 \cup (\cup_{j=0}^{k-1} H_j) - a(A_0 \setminus \cup_{j=1}^{k-1} H_j)$; we get for any $t \in [\theta_{k-1}, \theta_k[$:

$$c_B(t) = c_{k-1} + u_k(t - \theta_{k-1}). \quad (1)$$

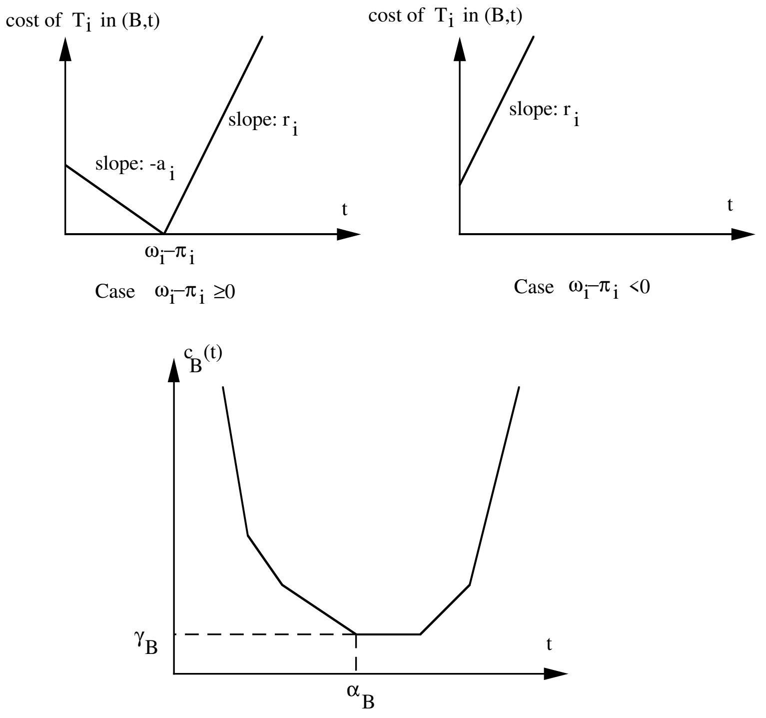

Figure 1: Cost functions

Let r be the number of terms of the sequence θ_k. Every task is tardy from time θ_r on. So for any $t \in [\theta_r, +\infty[$, we have:

$$c_B(t) = c_r + (t - \theta_r)r(B). \tag{2}$$

From (1), (2) and since $c_B(t)$ is a continuous time function on $[0, +\infty[$, we get that $c_B(t)$ is piecewise linear. Moreover the slopes $u_k, k \in \{1, \cdots, r\}$ of the successive pieces are strictly increasing since we have:

$$u_k - u_{k-1} = r(H_{k-1}) + a(H_{k-1}) > 0.$$

So $c_B(t)$ is a piecewise linear and convex time function. ∎

We derive from Property (1) that there is a *unique* time instant α_B such that:

$$\forall \epsilon > 0, \quad c_B(\alpha_B - \epsilon) > c_B(\alpha_B) \text{ and } c_B(\alpha_B + \epsilon) \geq c_B(\alpha_B).$$

In what follows, we denote by γ_B the value $c_B(\alpha_B)$. Figure 1 shows a pair of values (α_B, γ_B).
The following three operations on a schedule $S = ((B_1, s_1), \cdots, (B_b, s_b))$ will appear to be quite useful (see Figure 2):

- $LEFTSHIFT(S, B'_k, s_k, t)$, where B'_k is a prefix of B_k and $f_{k-1} < t < s_k$, is the schedule we get by left shifting (B'_k, s_k) until it becomes (B'_k, t);
- $RIGHTSHIFT(S, B"_k, s_k + p(B_k) - p(B"_k), t)$, where $B"_k$ is a suffix of B_k and $s_k + p(B_k) - p(B"_k) < t < s_{k+1}$, is the schedule we get by right-shifting $(B"_k, s_k + p(B_k) - p(B"_k))$ until it becomes $(B"_k, t)$;
- $LEFTSHIFT\&MERGE(S, B'_k, s_k)$, where B'_k is a prefix of B_k and $k > 1$, is the schedule we get by left shifting (B'_k, s_k) until it becomes (B'_k, f_{k-1});

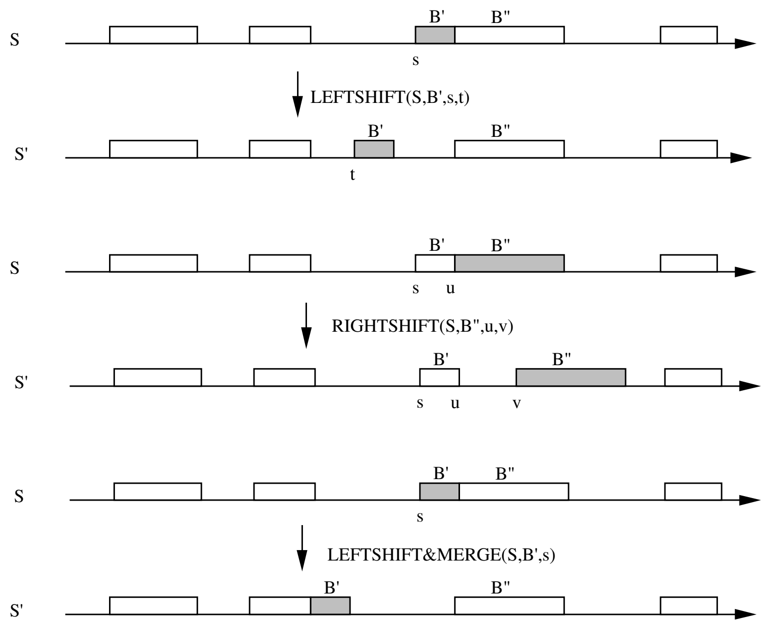

Figure 2: 3 basic operations on a schedule

Let $S = ((B_1, s_1), \cdots, (B_b, s_b))$ be a schedule. The allocated block (B_k, s_k) is said to be *left-adjusted* if for any time $t \in [f_{k-1}, s_k[$, the inequality $c_{B_k}(t) > c_{B_k}(s_k)$ (where by convention $f_0 = 0$) is satisfied. By extension, the schedule

S itself is said to be left-adjusted if all its allocated blocks are left-adjusted. The following property shows that there is an optimal schedule which is left-adjusted.

Property 2 *Left-adjusted schedules make a dominant subset.*

Proof. — Let $S = ((B_1, s_1), \cdots, (B_b, s_b))$ be an non left-adjusted *optimal* schedule. Since S is optimal, for any $k \in \{1, \cdots, b\}$, we have $\forall t \in [f_{k-1}, s_k[, c_{B_k}(t) \geq c_{B_k}(s_k)$. Since S is not left-adjusted, let k_0 be the first non left-adjusted alllocated block and let v be the *smallest* time in $[f_{k_0-1}, s_{k_0}[$ such that $c_{B_{k_0}}(t) = c_{B_{k_0}}(s_{k_0})$. We then define the schedule S' as follows.
If $v > f_{k_0-1}$ then $S' = LEFTSHIFT(S, B_{k_0}, s_{k_0}, v)$. From the definition of v, we know that the allocated block (B_{k_0}, v) of S' is left-adjusted.
If $v = f_{k_0-1}$ and $k_0 > 1$ then $S' = LEFTSHIFT\&MERGE(S, B_{k_0}, s_{k_0})$. Since (B_{k_0-1}, s_{k_0-1}) is left-adjusted in S and $c_{B_{k_0}}(s_{k_0}) = c_{B_{k_0}}(v)$ we get from Property 1 that the allocated block $(B_{k_0-1}B_{k_0}, s_{k_0-1})$ is left-adjusted in S'.
If $v = f_{k_0-1}$ and $k_0 = 1$ then $S' = LEFTSHIFT(S, B_1, s_1, 0)$.
Let us denote respectively by b' and k'_0 the number of allocated blocks and the index of the first non left-adjusted allocated block in S'. Whatever the case, we have $b' - k'_0 < b - k_0$. So, after iterating the process at most $b - k_0$ times we get an optimal and left-adjusted schedule. ∎

3 Symmetric and task-independent costs

Garey et al. have proposed in [1] an $O(n \log n)$ algorithm for the special case when for any task T_i, $a_i = r_i = 1$. This algorithm, that will be called GTW in the rest of the paper computes an optimal left-adjusted schedule S^2 of the restriction of the problem to its first $q + 1$ tasks from an optimal left-adjusted schedule S^1 of the restriction of the problem to its first q tasks as follows:

1. if $\omega_{q+1} > f_{b(S_1)}(S^1)$ then S^2 is got by creating the allocated block $((T_{q+1}), \omega_{q+1})$ and adding it to S^1;

2. if $\omega_{q+1} < f_{b(S_1)}(S^1)$ then let S the schedule we get by adding the task T_{q+1} as the last task of the last allocated block of S^1.
   If the last allocated block of S has less tardy tasks than on-time or early tasks then $S^2 = S$.

Otherwise the last allocated block of S is left-shifted until its starting time t matches one of the three following events:

E1: $t = 0$;

E2: the number of tardy tasks of the shifted block strictly decreases at time t;

E3: t is the completion time of the one-but-last block of S^1.

In case of event **E1** or **E2**, $S^2 = LEFTSHIFT(S, B_{b(S)}, s_{b(S)}, t)$; in case of event **E3**, $S^2 = LEFTSHIFT\&MERGE(S, B_{b(S)}, s_{b(S)})$.

The above GTW algorithm is illustrated on Figure 3 that shows the 6 first iterations associated with the following input data.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p_i	2	3	1	2	1	2	2	1	1	2	1	5	1	2	2
ω_i	4	1	7	8	5	3	13	14	16	18	19	15	16	17	18

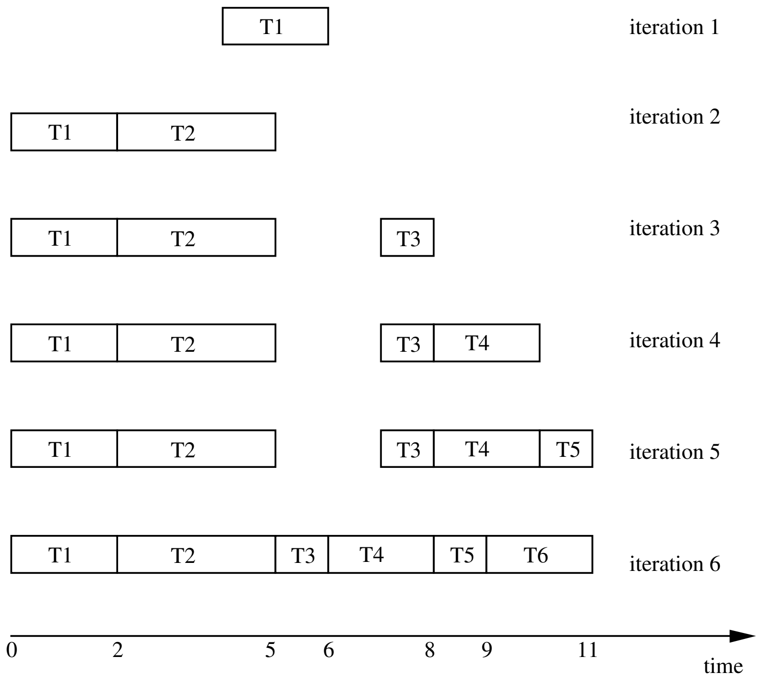

Figure 3: the GTW algorithm

The correctness of GTW mainly results from the following property whose proof is in [1]

Property 3 *The schedule provided by iteration k of GTW is a left-adjusted optimal schedule for the restriction of the problem to its k first tasks.*

In [1], the authors also note that their algorithm may be simply extended to the case when the execution cost of task T_i is $w_i c_i(t_i)$.

4 Asymmetric and task-independent costs

4.1 The EXT-GTW algorithm

This section proposes an extension of GTW called EXT-GTW for the case when for any task T_i, we have $a_i = a$ and $r_i = r$ where it is only assumed that a and r are non negative. Let (B, s) be an allocated block. The inequalities $LEFT(B, s)$ et $RIGHT(B, s)$ are defined by:

$LEFT(B, s)$: $a(A(B, s) + H(B, s)) - rR(B, s) \geq 0$;

$RIGHT(B, s)$: $r(R(B, s) + H(B, s)) - aA(B, s) \geq 0$.

where $A(B, s)$, $H(B, s)$ et $R(B, s)$ are respectively the number of early, on-time and tardy tasks in (B, s).
The algorithm EXT-GTW differs from GTW by the block invariant satisfied by all the allocated blocks at each iteration and by the fact that within each iteration EXT-GTW may *repeat the merging process* as long as the last allocated block of the running schedule does not satisfy the invariant. As for GTW, we describe the generic step of EXT-GTW that provides an optimal schedule S^2 of the restriction of the problem to its first $q + 1$ tasks from an optimal schedule S^1 of the restriction of the problem to its first q tasks.

1. if $\omega_{q+1} > f_{b(S_1)}(S^1)$ then S^2 is got by creating the allocated block $((T_{q+1}), \omega_{q+1})$ and adding it to S^1;

2. if $\omega_{q+1} < f_{b(S_1)}(S^1)$ then let S be the schedule we get by making task T_{q+1} be the last task of the last allocated block of S^1.
(a) If $LEFT(S_{b(S)})$ is true, then $S^2 = S$.
(b) Otherwise, the last allocated block of S is shifted to the left as long as its starting time t matches one of the three following events:

F1: $t = 0$;

F2: $LEFT(S_{b(S)})$ is true;

F3: t is the completion time of the one-but-last allocated block of S.

If **F1** or **F2** occurs, then $S^2 = LEFTSHIFT(S, B_{b(S)}, t)$. If **F3** occurs then $S := LEFTSHIFT\&MERGE(S, B_{b(S)}, s_{b(S)})$ and return to 2.(a).

An allocated block (B, s) is said to be *left-optimal* if for any (B', s) where B' is a prefix of B, the inequality $LEFT(B', s)$ is true. An allocated block (B, s) is said to be *right-optimal* if for any $(B", s+p(B)-p(B"))$ where $B"$ is a suffix of B, the inequality $RIGHT(B", s + p(B) - p(B"))$ is true. An allocated block (B, s) is said to be *quasi left-optimal* if $LEFT(B, s)$ is false and if for any (B', s) where B' is a proper prefix of B, the inequality $LEFT(B', s)$ is true. The following property gives a strong structural condition met by the optimal and left-adjusted schedules.

Property 4 *Let $S = ((B_1, s_1), \cdots, (B_b, s_b))$ be an optimal and left-adjusted schedule. Any allocated block (B_k, s_k) such that $s_k > 0$ is left and right optimal. Moreover if $s_1 = 0$ then the allocated block (B_1, s_1) is right-optimal.*

Proof. — Assume that $s_k > 0$ and that (B_k, s_k) is not left-optimal. There is a prefix B'_k of B_k such that $LEFT(B'_k, s_k)$ is false. There also exists a sufficiently small $\epsilon > 0$ such that:

1. $\mathcal{R}(B'_k, s_k - \epsilon) = \mathcal{R}(B'_k, s_k)$,

2. the schedule $S' = LEFTSHIFT(B'_k, s_k, s_k - \epsilon)$ is feasible.

From the definition of ϵ we have:

$$c(S') = c(S) + \epsilon(a(A(B'_k, s_k) + H(B'_k, s_k)) - rR(B'_k, s_k))$$

since the tardy tasks of $(B'_k, s_k - \epsilon)$ are the tardy tasks of (B'_k, s_k), the early tasks of $(B'_k, s_k - \epsilon)$ are the early or on-time tasks in (B'_k, s_k) and there is no on-time task in $(B'_k, s_k - \epsilon)$. As $LEFT(B'_k, s_k)$ is false, we have $c(S') < c(S)$, what contradicts the optimality of S.
Assume that $s_k > 0$ and that (B_k, s_k) is not right-optimal. Let $u_k = s_k + p(B_k) - p(B"_k)$. There is a suffix $B"_k$ de B_k such that $RIGHT(B"_k, u_k)$ is false. There also exists a sufficiently small $\epsilon > 0$ such that:

1. $\mathcal{A}(B"_k, u_k + \epsilon) = \mathcal{A}(B"_k, u_k)$,

2. the schedule $S" = RIGHTSHIFT(B"_k, u_k, u_k + \epsilon)$ is feasible.

From the definition of ϵ we have:

$$c(S") = c(S) + \epsilon(r(R(B"_k, u_k) + H(B"_k, u_k)) - aA(B"_k, u_k))$$

since the early tasks of $(B"_k, u_k + \epsilon)$ are the early tasks of $(B"_k, u_k)$, the tardy tasks of $(B"_k, u_k + \epsilon)$ are the tardy or on-time tasks of $(B"_k, u_k)$ and there is no on-time task in $(B"_k, u_k - \epsilon)$. As $RIGHT(B"_k, u_k)$ is false, we have $c(S") < c(S)$, what contradicts the optimality of S.
If $s_1 = 0$, the same argument as before applied to a suffix of B_1 yields a contradiction to the optimality of S if the allocated block (B_1, s_1) is not right-optimal. ∎

We now prove a dominance property of the left-optimal allocated blocks, a symmetric property of the right-optimal allocated blocks and a theorem that more generally applies to the right and left optimal allocated blocks.

Theorem 1 *Let (B, s) be a left-optimal allocated block. The cost of any schedule of B whose last task completes at most at time $f = s + p(B)$ is not less than the cost of (B, s).*

Proof. — Let us assume that $B = (T_1, \cdots, T_n)$. Let σ be an arbitrary schedule of B whose last task completes at most at time f. Let u_i be the starting time of T_i in (B, s), v_i be the starting time of T_i in σ and $\Delta_i = u_i - v_i$. From the assumptions on σ we derive that for any $i \in \{1, \cdots, n\}$, $\Delta_i \geq 0$ and

$$\Delta_n \leq \cdots \leq \Delta_1.$$

If T_i is early or on-time in (B, s), its cost in σ is exactly $a\Delta_i$ larger than in (B, s), otherwise T_i is tardy in (B, s) and its cost in σ is *at most* $r\Delta_i$ less than in (B, s). So if c_1 is the cost of (B, s) and c_2 the cost of σ, we have:

$$c_2 \geq c_1 + a(\sum_{T_i \in \mathcal{A}(B,s) \cup \mathcal{H}(B,s)} \Delta_i) - r(\sum_{T_i \in \mathcal{R}(B,s)} \Delta_i)$$

We thus have to prove that:

$$a(\sum_{T_i \in \mathcal{A}(B,s) \cup \mathcal{H}(B,s)} \Delta_i) - r(\sum_{T_i \in \mathcal{R}(B,s)} \Delta_i) \geq 0 \qquad (3)$$

For any $k \in \{1, \cdots, n\}$, let us denote by B_k the prefix $(T_1, \cdots, T_k)$, by $A_k = \{T_{i_1}, \cdots, T_{i_{a_k}}\}$ the subset of early or on-time tasks in (B_k, s) and by

$R_k = \{T_{j_1}, \cdots, T_{j_{r_k}}\}$ the subset of the tardy tasks in (B_k, s). Without loss of generality, we assume that

$$i_1 < \cdots < i_{a_k} \text{ and } j_1 < \cdots < j_{r_k}.$$

Let $\alpha_k = \max_{j\in\{1,\cdots,k\}}\{\frac{r_j}{a_j}\}$ and let k^* *the smallest* index in $\{1, \cdots, k\}$ such that $\frac{r_j}{a_j} = \alpha_k$. Notice that α_k is well-defined for any $k \in \{1, \cdots, n\}$: indeed we have $aa_1 - rr_1 \geq 0$ since (B, s) is left-optimal and $a_1 + r_1 = 1$. We thus get that $a_1 = 1$ and $r_1 = 0$ (T_1 is early in (B, s)), from which we conclude that $a_k > 0$ for any $k \in \{1, \cdots, n\}$.
Let us define by $\mathcal{T}_k$ the following transportation problem:

- A_k is the set of suppliers and the availability of each supplier is r_{k^*},
- R_k is the set of demands and the amount of each demand is a_{k^*},
- the demands must be exactly fullfilled,
- a transportation arc $(T_i, T_j) \in A_k \times R_k$ is feasible if $i < j$.

Such a transportation program is shown in Figure 4.

The following property shows that the transportation problems $\mathcal{T}_k$ are feasible.

Propriété 1 *For any $k \in \{1, \cdots, n\}$, the problem $\mathcal{T}_k$ is feasible.*

Proof. — $\mathcal{T}_1$ is feasible since $R_1 = \emptyset$.
Assume now that Σ_k is a feasible solution of $\mathcal{T}_k$ and let us consider the two following cases about the feasiblity of $\mathcal{T}_{k+1}$ depending on whether T_{k+1} is an early or on-time task or a tardy task in (B, s).

First case: $T_{k+1} \in A_{k+1}$.
We then have $(k+1)^* = k^*$ and problem $\mathcal{T}_{k+1}$ has one more supplier (line $a_k + 1$) than $\mathcal{T}_k$. Since the availability of the suppliers are the same in $\mathcal{T}_k$ and in $\mathcal{T}_{k+1}$, Σ_k is also feasible solution for $\mathcal{T}_{k+1}$.

Second case: $T_{k+1} \in R_{k+1}$.
Let us consider two subcases depending on whether $(k+1)^* = k^*$ or $(k+1)^* = k+1$.
First subcase: $(k+1)^* = k^*$.
We then have

$$\frac{r_{k+1}}{a_{k+1}} = \frac{r_k + 1}{a_k} \leq \frac{r_{k^*}}{a_{k^*}}$$

	T4	T7	T9	T10	T11	T13	T14	T15	T16	T18	T19	T20	T21
T1	8	5											
T2		3	8	2									
T3				6	7								
T5					1	8	4						
T6							4	8	1				
T8									7	6			
T12										2	8	3	
T17												5	8

Bk=(T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13,T14,T15,T16,T17,T18,T19,T20,T21)
Ak=(T1,T2,T3,T5,T6,T8,T12,T17)
Rk=(T4,T7,T9,T10,T11,T13,T14,T15,T16,T18,T19,T20,T21)
availability of each task in Ak: 13;
demand of each task in Bk: 8;
fordidden cells in grey;
α(k)=13/8; k*=21.

Figure 4: A transportation program $\mathcal{T}_k$

$\mathcal{T}_{k+1}$ has one more demand (column r_k+1) than $\mathcal{T}_k$. The availability of the suppliers and the amounts of the demands are the same in $\mathcal{T}_k$ and in $\mathcal{T}_{k+1}$. Since on one hand all the cells of the last column of $\mathcal{T}_{k+1}$ are feasible and on the other hand the difference $r_{k^*}a_k - a_{k^*}r_k$ between the total avalailability and the total demand of $\mathcal{T}_k$ is at least a_{k^*} from the above inequality, Σ_k may be extended into a feasible solution Σ_{k+1} of $\mathcal{T}_{k+1}$.
Second subcase: $(k+1)^* = k+1$.
In $\mathcal{T}_{k+1}$, the availability of each of the a_k suppliers is r_k+1 and the amount of each of the r_k+1 demands is a_k. So, from the definition of k^* we get:

$$\forall j \in \{1, \cdots, k\}, \quad \frac{r_j}{a_j} < \frac{r_k+1}{a_k} \tag{4}$$

Assume that $1 + r_k = q_k a_k + \rho_k$ where $0 \leq \rho_k < a_k$. We then build *line by line* a solution Σ_{k+1} of $\mathcal{T}_{k+1}$ as follows:

- the q_k+1 first cells of the first line are respectively $a_k, \cdots, a_k, \rho_k$ whereas the other cells of that line are null; we then define $c(1) = q_k+1$ and $\rho(1) = \rho_k$.

- assume that the $l-1$ first lines of $\mathcal{T}_{k+1}$ are built;
  If $\rho(l-1)+\rho_k \leq a_k$ then the values of the q_k+1 cells whose column numbers are $c(l-1), \cdots, c(l-1)+q_k$ are respectively

  $$a_k - \rho(l-1), a_k, \cdots, a_k, \rho(l-1)+\rho_k$$

  wheras the other cells of line l are null; we then define $c(l) = c(l-1)+q_k$ and $\rho(l) = \rho(l-1)+\rho_k$.
  If $\rho(l-1)+\rho_k > a_k$ then the values of the q_k+2 cells whose column numbers are $c(l-1), \cdots, c(l-1)+q_k+1$ are respectively

  $$a_k - \rho(l-1), a_k, \cdots, a_k, \rho(l-1)+\rho_k - a_k$$

  whereas the other cells of line l are null; we then define $c(l) = c(l-1)+q_k+1$ and $\rho(l) = \rho(l-1)+\rho_k - a_k$.

The lines containing $q_k - 1$ (respectively q_k) intermediate cells with value a_k are said to be of type 1 (respectively type 2). The following invariant is easily verified:

Property 5 *When line l is built, the demands of columns* 1 *to* $c(l)-1$ *are satisfied and* $0 \leq \rho(l) \leq a_k$.

On the example of Figure 4, we have $r_k{+}1 = 13$, $a_k = 8$, $(k{+}1)^* = k{+}1 = 21$, $\alpha_{k+1} = \frac{13}{8}$, $q_k = 1$ and $\rho_k = 5$. The solution built for $\mathcal{T}_{21}$ in written in the transportation array. Lines $1, 3, 6, 8$ are of type 1, lines $2, 4, 5, 7$ are of type 2.

Σ_{k+1} is a feasible solution of $\mathcal{T}_{k+1}$ if and only if its non-zero valued cells are feasible cells. We first show that there are exactly r_k+1 columns with at least one non-zero cell and we prove next that every non-zero valued cell of Σ_{k+1} is a feasible cell.

Let C be the number of columns with a non-zero valued cell in Σ_{k+1}. From the definition of Σ_{k+1}, the demands of the $C-1$ first columns are exactly fullfilled whereas the last column receives $\rho(a_k)$. Since in Σ_{k+1} each of the a_k suppliers sends its whole availability $1+r_k$, we have:

$$(1+r_k)a_k = (C-1)a_k + \rho(a_k)$$

Since $0 \leq \rho(a_k) \leq a_k$, the previous inequality implies $\rho(a_k) = a_k$ and $C = 1+r_k$.

Let us call the line separating the feasible cells from the unfeasible cells

of the transportation array the *borderline* F of $\mathcal{T}_{k+1}$ (see Figure 4). For any line l, let y_l be the greatest column number such that the point with coordinates (l, y_l) in the transportation array belongs to F.
The following property shows that the non-zero cells in Σ_{k+1} are feasible cells of $\mathcal{T}_{k+1}$.

Property 6 *For any line $l \in \{1, \cdots, a_k - 1\}$, we have $y_l \leq c(l) - 1$.*

Proof. — Let us consider the first line. We have $c(1) = q_k + 1$. The point $(1, y_1)$ on F is associated with a prefix B_{j_1} such that $r_{j_1} = y_1$ and $a_{j_1} = 1$. We then get from (4) that:

$$y_1 < q_k + \frac{\rho_k}{a_k} \leq q_k + 1$$

The first line thus satisfies the property.
Assume now that among the l first lines, there are l_1 lines of type 1 and l_2 lines of type 2. We then have $\rho(l) = l_1\rho_k + l_2(\rho_k - a_k)$.
If

$$\rho(l) + a_k = (l_1 + 1)\rho_k + l_2(\rho_k - a_k) \leq a_k \tag{5}$$

then the line $l+1$ is of type 1 and we have $c(l+1) = (l+1)q_k + l_2 + 1$. The point $(l + 1, y_{l+1})$ on F corresponds to a prefix $B_{j_{l+1}}$ such that $r_{j_{l+1}} = y_{l+1}$ and $a_{j_1} = l + 1$. Since from (4) we have:

$$\frac{y_{l+1}}{l+1} < q_k + \frac{\rho_k}{a_k}$$

We get from (5) that $y_{l+1} < (l+1)q_k + l_2 + 1$ and thus that $y_{l+1} \leq c(l+1) - 1$.
If

$$\rho(l) + a_k = (l_1 + 1)\rho_k + l_2(\rho_k - a_k) > a_k \tag{6}$$

then line l is of type 2 and we have $c(l + 1) = (l + 1)q_k + l_2 + 2$. The point $(l + 1, y_{l+1})$ on F corresponds to a prefix $B_{j_{l+1}}$ such that $r_{j_{l+1}} = y_{l+1}$ and $a_{j_1} = l + 1$. From (4) we get:

$$\frac{y_{l+1}}{l+1} < q_k + \frac{\rho_k}{a_k}$$

Since $\rho(l) + a_k \leq 2a_k$, we have $(l + 1)\rho_k - l_2 a_k \leq 2a_k$, which rewrites

$$(l+1)\frac{\rho_k}{a_k} \leq l + 2$$

We thus have $y_{l+1} < (l+1)q_k + l_2 + 2$ and $y_{l+1} \leq c(l+1) - 1$. That completes the proof of Property 6. ∎

To conclude the proof of Property 1, notice that if Σ_{k+1} is not feasible, there necessarily exists a line $l \in \{1, \cdots, a_k - 1\}$ such that $y_l \geq c(l)$. We thus get from 6 that Σ_{k+1} is a feasible solution. ∎

Recall that in order to prove Theorem 1, we have to prove the inequality (3):

$$a\Big(\sum_{T_i \in \mathcal{A}(B,s) \cup \mathcal{H}(B,s)} \Delta_i\Big) - r\Big(\sum_{T_i \in \mathcal{R}(B,s)} \Delta_i\Big) \geq 0$$

where $B = (T_1, \cdots, T_n)$ and (B, s) is a left-optimal allocated block. Since (B, s) is left-optimal, we have $aa_{n^*} - rr_{n^*} \geq 0$. A sufficient condition for (3) is:

$$r_{n^*}\Big(\sum_{T_i \in A_n} \Delta_i\Big) - a_{n^*}\Big(\sum_{T_i \in B_n} \Delta_i\Big) \geq 0. \tag{7}$$

But from Property 1, we know that $\mathcal{T}_n$ has a *feasible solution* $n_{l,c}, l \in \{1, \cdots, a_n\}, c \in \{1, \cdots, r_n\}$. Let $J(l)$ (respectively $I(c)$) the column (respectively line) numbers associated with a feasible cell of line l (respectively column c). We have:

$$\begin{array}{ll}
\forall l \in \{1, \cdots, a_n\} & \sum_{c \in J(l)} n_{l,c} \leq r_{n^*} \\
\forall c \in \{1, \cdots, r_n\} & \sum_{l \in I(c)} n_{l,c} = a_{n^*} \\
\forall (l,c), l \in \{1, \cdots, a_n\}, c \in J(l) & \Delta_{i_l} \geq \Delta_{j_c} \\
\forall (l,c), l \in \{1, \cdots, a_n\}, c \notin J(l) & n_{l,c} = 0
\end{array}$$

For any line $l \in \{1, \cdots, a_n\}$, we thus have:

$$r_{n^*} \Delta_{i_l} \geq \sum_{c \in J(l)} n_{l,c} \Delta_{j_c}.$$

By summing all these inequalities we get:

$$r_{n^*} \sum_{l=1}^{a_n} \Delta_{i_l} \geq \sum_{c=1}^{r_n} \Big(\Delta_{j_c} \sum_{l \in I(c)} n_{l,c}\Big)$$

which rewrites:

$$r_{n^*} \sum_{l=1}^{a_n} \Delta_{i_l} \geq a_{n^*} \sum_{c=1}^{r_n} \Delta_{j_c}.$$

Since the inequality (7) is satisfied, the same is true for inequality (3), what completes the proof of Theorem 1. ∎

The right-optimal allocated blocks satisfy the following symmetrical property.

Theorem 2 *Let (B, s) be a right-optimal allocated block. The cost of any schedule of B whose first task starts at least at time s is not less than the cost of (B, s).*

Proof. — Since that proof is quite similar to the proof of Theorem 1, we only give its global scheme. Let $B = (T_n, \cdots, T_1)$ and let τ be an arbitrary schedule of B whose first task starts at least at time s. Let u_i and v_i be respectively the starting times of task T_i in (B, s) and in τ. From the assumptions on τ we get that $\Delta_i = v_i - u_i \geq 0$ and that

$$\Delta_n \leq \cdots \leq \Delta_1.$$

Let c_1 and c_2 be respectively the costs of (B, s) and τ. If T_i is on-time or tardy in (B, s), its cost in τ is exactly $r\Delta_i$ larger, otherwise if it is early in (B, s), its cost in τ is at most $a\Delta_i$ less. We thus have:

$$c_2 \geq c_1 + r(\sum_{T_i \in \mathcal{R}(B,s) \cup \mathcal{H}(B,s)} \Delta_i) - a(\sum_{T_i \in \mathcal{A}(B,s)} \Delta_i)$$

and we must prove that:

$$r(\sum_{T_i \in \mathcal{R}(B,s) \cup \mathcal{H}(B,s)} \Delta_i) - a(\sum_{T_i \in \mathcal{A}(B,s)} \Delta_i) \geq 0 \tag{8}$$

For each $k \in \{1, \cdots, n\}$, let B_k be the suffix $(T_k, \cdots, T_1)$, $R_k = \{T_{i_1}, \cdots, T_{i_{r_k}}\}$ be the set of the early tasks on-time and tardy tasks of $(B_k, s + \sum_{i=k+1}^{n} p_i)$ and $A_k = \{T_{j_1}, \cdots, T_{j_{a_k}}\}$ be the set of the early tasks of $(B_k, s + \sum_{i=k+1}^{n} p_i)$. Without any loss of generality we assume that

$$i_1 < \cdots < i_{r_k} \text{ et } j_1 < \cdots < j_{a_k}.$$

Let $\beta_k = \max_{j \in \{1, \cdots, k\}} \{\frac{a_j}{r_j}\}$ and $\hat{k}$ be *the smallest* index of $\{1, \cdots, k\}$ such that $\frac{a_j}{r_j} = \beta_k$. We notice that β_k is defined for each $k \in \{1, \cdots, n\}$: indeed we have $a_1 + r_1 = 1$ and $rr_1 - aa_1 \geq 0$ since (B, s) is right-optimal, we thus have $r_1 = 1$ et $a_1 = 0$ (T_1 is tardy), from which we get that $r_k > 0$ for any $k \in \{1, \cdots, n\}$.

Let us define the transportation problem $\mathcal{U}_k$ where:

- R_k is the set of suppliers and the availability of each supplier is $a_{\hat{k}}$,

- A_k is the set of demands and the amount of each demand is $r_{\hat{k}}$,
- the demands must be exactly fullfilled,
- a transportation arc $(T_i, T_j) \in A_k \times R_k$ is feasible if $i < j$.

We then have the symmetrical property of Property 1 whose proof, which is analog to that of Property 1 is omitted:

Property 7 *For any $k \in \{1, \cdots, n\}$, $\mathcal{U}_k$ has a feasible solution.*

Since (B, s) is right-optimal, we have

$$rr_{\hat{n}} - aa_{\hat{n}} \geq 0.$$

A sufficient condition for (8) to be satisfied is that:

$$a_{\hat{n}}(\sum_{T_i \in R_n} \Delta_i) - r_{\hat{n}}(\sum_{T_i \in A_n} \Delta_i) \geq 0. \qquad (9)$$

But the feasibility of $\mathcal{U}_n$ implies that (9) is satisfied, what completes the proof of 2. ∎

Theorems 1 and 2 may be generalized to left and right optimal allocated blocks as follows:

Theorem 3 *Let (B, s) be a left and right optimal allocated block. The cost of (B, s) is at most the cost of an arbitrary schedule of B.*

Proof. — Let σ be an arbitrary schedule of B. If σ completes at most at time $s + p(B)$ (respectively starts at least at time s), Theorem 1 (respectively 2) shows $c_B(s) \leq c(\sigma)$. Otherwise, there is a prefix B' of B such that the last task of B' is completed in σ at most at time $s + p(B')$ and such that the first task of the compementary suffix $B"$ of B' in B is started at least at time $s + p(B')$. Let σ' (respectively $\sigma"$) be the restriction of σ to the tasks of B' (respectively $B"$). Since (B', s) is left-optimal, we have $c_{B'}(s) \leq c(\sigma')$ from Theorem 1. Since $(B", s + p(B) - p(B"))$ is right-optimal, we get from Theorem 2 that $c_{B"}(s + p(B) - p(B")) \leq c(\sigma")$. We thus may conclude that $c_B(s) \leq c(\sigma)$. ∎

4.2 Correctness of EXT.GTW

We show in this section that if S^1 is a left-adjusted optimal schedule for the restriction of the problem to its first q tasks, then the schedule S^2 provided by the generic step of EXT.GTW is also a left-adjusted optimal schedule for the restriction of the problem to its first $q+1$ first tasks.
In order to prove the correctness of EXT.GTW, we assume that for any $k \in \{1, \cdots, b(S^1)\}$, the restriction $S^1[1, .., n_k(S^1)]$ of S^1 to its k first allocated blocks is optimal and left-adjusted for the task sequence $(T_1, \cdots, T_{n_k(S^1)})$ and we show the same is true for S^2.
If $\omega_{q+1} < f_{b(S^1)}(S^1)$, let $\hat{S}^0$ be the first schedule built by the generic step of EXT.GTW by adding task T_{q+1} to the last allocated block of S^1. Then if EXT.GTW performs K mergings, let $\hat{S}^1, \cdots, \hat{S}^K$ the intermediate schedules we got just after these mergings. Notice that if $K \geq 1$, then S^2 is either $\hat{S}^K$ or results from the occurrence of event **F1** or event **F2** during the left shift of the last allocated block of $\hat{S}^K$. Let Σ be an *arbitrary* schedule for the tasks sequence $(T_1, \cdots, T_{q+1})$. Before we examine the different issues of the generic step of EXT.GTW, we give two properties that will simplify the proof: the first shows that an allocated block remains right-optimal and quasi left-optimal when it is left-shifted as long as the initially tardy tasks remain tardy.

Property 8 *Let (D, u) be a right optimal and quasi left-optimal allocated block and let $v < u$. If $\mathcal{R}(D, v) = \mathcal{R}(D, u)$ then (D, v) is right optimal and quasi left-optimal.*

Proof. — Let A_1, H_1, R_1 (respectively A_2, H_2, R_2) the number of early, on-time and tardy tasks in (D, u) (respectively (D, v)). Since each tardy task of (D, u) is still tardy in (D, v), we have $A_2 = A_1 + H_1$, $H_2 = 0$ and $R_2 = R_1$. Since $LEFT(D, u)$ is false, we have $a(A_1 + H_1) < rR_1$ and so we get

$$a(A_2 + H_2) < rR_2 \tag{10}$$

which implies that $LEFT(D, v)$ is false.
Consider a proper prefix D' of D and let A'_1, H'_1, R'_1 (respectively A'_2, H'_2, R'_2) be the number of early, on-time and tardy tasks in (D', u) (respectively (D', v)). Since any tardy task in (D', u) is still tardy in (D', v), we get: $A'_2 = A'_1 + H'_1$, $H'_2 = 0$ and $R'_2 = R'_1$. As $LEFT(D', u)$ is true, we have $a(A'_1 + H'_1) \geq rR'_1$ and so $a(A'_2 + H'_2) \geq rR'_2$, which implies that $LEFT(D', v)$ is true. Thus (D, v) is quasi left-optimal.

Let $D"$ be a proper suffix of D. We denote by $A"_1$, $H"_1$ and $R"_1$ (respectively $A"_2$, $H"_2$ and $R"_2$) the number of early, on-time and tardy tasks in $(D", u + p(D) - p(D"))$ (respectively $(D", v + p(D) - p(D")))$. Let $\hat{D}$ be the complementary proper prefix of $D"$ in D. Since each tardy task in $(\hat{D}, u)$ is still tardy in $(\hat{D}, v)$, we have: $\hat{A}_2 = \hat{A}_1 + \hat{H}_1$, $\hat{H}_2 = 0$ and $\hat{R}_2 = \hat{R}_1$. As $LEFT(\hat{D}, v)$ is true, we have $a(\hat{A}_2 + \hat{H}_2) \geq r\hat{R}_2$, which rewrites $a(-A_2 + A"_2 - H_2 + H"_2) \leq r(-R_2 + R"_2)$. By summing that inequality to inequality (10), we get $a(A"_2 + H"_2) > rR"_2$, which implies $r(R"_2 + H"_2) > aA"_2$ and so $RIGHT(D", v + p(D) - p(D"))$ is true.
For the suffix D itself, inequality (10) directly implies that $r(R_2 + H_2) > aA_2$ and so $RIGHT(D, v)$ is true. (D, v) is thus a right-optimal allocated block.

■

The second property whose simple proof is omitted concerns the merging of a right-optimal and quasi left-optimal allocated block with a left and right-optimal allocated block.

Property 9 *Let (E, u) be a left and right-optimal allocated block and let $(D, u + p(E))$ a right-optimal and quasi left-optimal allocated block. The allocated block (ED, u) is right-optimal. Moreover, (ED, u) is also left-optimal (respectively quasi left-optimal) if $LEFT(ED, u)$ is true (respectively false).*

We now analyze the different issues of the generic step of EXT.GTW.

Case 1: $\omega_{q+1} > f_{b(S^1)}(S^1)$

We have $c(S^2) = c(S^1)$ since the cost of the last allocated block of S^2 is zero and since $c(S^1) \leq c(\Sigma[1, .., q])$ from the induction. Since $c(\Sigma) \geq c(\Sigma[1, .., q])$, we have $c(\Sigma) \geq c(S^2)$. So S^2 is optimal and left-adjusted.

Case 2: $\omega_{q+1} < f_{b(S^1)}(S^1)$ et $LEFT(\hat{S}^0_{b(\hat{S}^0)})$ is true

If the last allocated block of S^1 does not start at time 0, then from the induction and Property 4, that allocated block is left and right-optimal. It is then easy to verify that the last allocated block of $\hat{S}^0$ which is obtained from the last allocated block of S^1 by adding the tardy task T_{q+1} is left and right-optimal too. Let l be the index of the last task of the last-but-one allocated block of S^1. From the induction, we have $c(\hat{S}^0[1, .., l]) = c(S^1[1, .., l] \leq c(\Sigma[1, .., l]$. Moreover we get from Theorems 1,2 and 3 that the cost $c[\hat{S}^0[l + 1, .., q + 1]$ of the last allocated block of $\hat{S}^0$ is at most equal

to $c(\Sigma[l+1,..,q+1]$. We thus have $c(\hat{S}^0) \leq c(\Sigma)$.
If the last allocated block of S^1 starts at time 0 (indeed S^1 has exactly one allocated block), then from the induction and Property 4, this bloc is right-optimal. Since the single allocated block making $\hat{S}^0$ is also right-optimal, we get from Theorem 3 that $c(\hat{S}^0) \leq c(\Sigma)$.

Cas 3: $\omega_{q+1} < f_{b(S^1)}(S^1)$ et $LEFT(\hat{S}^0_{b(\hat{S}^0)})$ is false

For each $k \in \{0, \cdots, K\}$, let (D_k, t_k) be the last allocated block of the intermediate schedule $\hat{S}^k$ and let respectively a_k, h_k, r_k be the number of early, on-time and tardy tasks of (D_k, t_k).
From Property 9, the allocated block (D_0, t_0) is right-optimal and quasi left-optimal since on the one hand $LEFT(D_0, t_0)$ is false and on the other hand this block results from adding to the last allocated block of S^1 (which from the induction is left and right-optimal) task T_{q+1}, which is a right-optimal and quasi left-optimal allocated block.
For any $k \in \{1, \cdots, K-1\}$, the allocated block (D_k, t_k) is right-optimal and quasi left-optimal since on the one hand $LEFT(D_k, t_k)$ is false and on the other hand this block results from left-shifting (D_{k-1}, t_{k-1}) (under the assumptions of Property 8) and the merging of an allocated block S^1 (which from the induction is left and right optimal with the allocated block (D_{k-1}, v) (where $v < t_{k-1}$), which is from Property 8, a right-optimal and quasi left-optimal allocated block.
Let us now consider the allocated block (D_K, t_K). Properties 8 and 9 yield that this block is right and left-optimal (respectively right and quasi left optimal) if $LEFT(D_K, t_K)$ is true (respectively false).
If $LEFT(D_K, t_K)$ is true, then $S^2 = \hat{S}^K$. If p is the number of tasks in D_K, we have $S^2[1,..,q+1-p] = S^1[1,..,q+1-p]$. From the induction, we have $c(S^1[1,..,q+1-p]) \leq c(\Sigma[1,..,q+1-p])$. Moreover since the last allocated block of S^2 is right and left-optimal, Theorem 3 implies that $c(S^2[q-p+2,..,q+1]) \leq c(\Sigma[q-p+2,..,q+1])$.
If $LEFT(D_K, t_K)$ is false, then there is one more block left-shifting that completes by the occurrence of event **F1** or **F3**.
If **F1** occurs, the associated left-shifting matches the assumptions of Property 8 and S^2 has a single allocated block that starts at time 0 and is right-optimal. We then get from Theorem 2 that $c(S^2) \leq c(\Sigma)$.
If **F2** occurs, the last allocated block of S^2 results from the left-shifting of (D_K, t_K) but this shift stops because the number of tardy tasks of the shifted allocated block strictly decreases.

Notice first that for any $k \in \{0, \cdots, K\}$, the inequality $a(a_k+h_k)-r(r_k-1) \geq 0$ is true. Indeed it is true by the definition of a_0, h_0 and r_0 for $k=0$. Let us assume it is true at the end of the $(k-1)^{th}$ merging and let a', h' and r' be respectively the number of early, on-time and tardy tasks of the allocated block of S^1 that is merged during the k^{th} merging. We then have $h_k = h'$, $a_k = a_{k-1} + h_{k-1} + a'$ and $r_k = r_{k-1} + r'$. Since the merged allocated block of S^1 is left-optimal we have $a(a'+h') - rr' \geq 0$ and since from the induction we have $a(a_{k-1} + h_{k-1}) - r(r_{k-1} - 1) \geq 0$, we get by summing these two inequalities $a(a_k + h_k) - r(r_k - 1) \geq 0$.
We thus have:

$$a(a_K + h_K) - r(r_K - 1) \geq 0. \quad (11)$$

Let (D_K, v) (where $v < t_K$) be the last allocated block of S^2 and let A, H et R be respectively the number of early, on-time and tardy tasks of (D_K, v). Assume that $x \geq 1$ tasks that are tardy in (D_K, t_K) are on-time in (D_K, v). The allocated block (D_K, v) itself satisfies $A = a_K + h_K$, $H = x$ and $R = r_K - x$. From Property (11) we get

$$a(A+H) - rR = a(a_K + h_K) - r(r_K - 1) + ax + r(x-1).$$

Since $x \geq 1$, $LEFT(D_K, v)$ is true. Moreover since $LEFT(D_K, t_K)$ is false, we have that $r(R+H) - aA = rr_K - a(a_K + h_K)$ is strictly positive, what implies that $RIGHT(D_K, v)$ is true too.
Let D' be a proper prefix of D_K. Let A'_1, H'_1, R'_1 be respectively the number of early, on-time and tardy tasks of (D', t_K) and let A'_2, H'_2, R'_2 be respectively the number of early, on-time and tardy tasks of (D', v). Let $y \geq 0$ be the number of tardy tasks of (D', t_K) that are on-time in (D', v). We have $A'_2 = A'_1$, $H'_2 = H'_1 + y$ and $R'_2 = R'_1 - y$. We thus get that

$$a(A'_2 + H'_2) - rR'_2 = a(A'_1 + H'_1) - rR'_1 + y(r+a)$$

what shows that $LEFT(D', v)$ is true.
Let $D"$ be a proper suffix of D_K. Let $A"_1, H"_1, R"_1$ be respectively the number of early, on-time and tardy tasks of $(D", t_K + p(D_K) - p(D"))$ and let $A"_2, H"_2, R"_2$ be respectively the number of early, on-time and tardy tasks of $(D", v + p(D_K) - p(D"))$. Let $z \geq 0$ be the number of tardy tasks in $(D", t_K + p(D_K) - p(D"))$ that are on-time in $(D", v + p(D_K) - p(D"))$. We have $A"_2 = A"_1$, $H"_2 = H"_1 + y$ and $R"_2 = R"_1 - y$. So we get that

$$r(R"_2 + H"_2) - aA"_2 = a(R"_1 + H"_1) - rR"_1$$

what shows that $RIGHT(D", v + p(D_K) - p(D"))$ is true.
As a conclusion the allocated block (D_K, v), which is the last allocated block of S^2 is left and right-optimal. If that block has p tasks, we have $S^2[1, .., q + 1 - p] = S^1[1, .., q+1-p]$. From the induction, we have $c(S^1[1, .., q+1-p]) \leq c(\Sigma[1, .., q + 1 - p])$ and from Theorem 3 we get $c(S^2[q - p + 2, ..q + 1]) \leq c(\Sigma[q - p + 2, ..q + 1])$. We thus may conclude that $c(S^2) \leq c(\Sigma)$.
We have shown that, for each issue of the generic step of EXT.GTW, S^2 is an optimal schedule for the tasks sequence $(T_1, \cdots, T_{q+1})$. That schedule is left-adjusted since on the one hand each allocated block, which is not the last one and that does not starts at time 0 is left-optimal from the induction and on the other hand we have shown that the last allocated block is also left-optimal for all issues of the generic step except event **F1**. Finally the restriction $S^2[1, .., n_k(S^2)]$ of S^2 to its k first allocated blocks is optimal and left-adjusted for the tasks sequence $(T_1, \cdots, T_{n_k(S^2)})$ since on the one hand that is true from the induction for $k \in \{1, \cdots, b(S^2) - 1\}$ and on the other hand we have shown that is also true for S^2 itself.
Since the generic step of EXT.GTW correct, EXT.GTW is also correct because the schedule S^1 provided for the single task T_1 is optimal, left-ajusted and has a single block.

4.3 Worst-case complexity of EXT.GTW

Let us associate with each allocated block (B_k, s_k) of the running schedule the heap T_k that contains the tardy tasks of (B_k, s_k), each with a priority equal to its tardiness. Each iteration of the mergings loop within the generic step of EXT.GTW performs a left-shifting whose complexity is $O(1)$ since it corresponds to add a constant to the prioriry of all the tasks in the heap and the merging that may be executed in $O(\log n)$). The key point here is to notice that the total number of mergings during an execution of EXT.GTW is $O(n)$ since each merging decreases by one the number of allocated blocks in the current schedule of EXT.GTW. The complexity of all the mergings is thus $O(n \log n)$. Apart from the merging loop, the complexity of the generic step of EXT.GTW is $O(1)$ except when the task T_{q+1} has to be inserted in the heap associated with the last allocated block of S^1 as its last (tardy) task, in which case the complexity is $O(logn)$. The overall worst-case complexity of EXT.GTW is thus $O(n \log n)$.

5 Asymmetric and task-dependent costs

The approach of Section 4 does not easily extend to the general problem where asymmetric and task-dependent costs are assumed. We present for that problem a polynomial algorithm based on the *convexity* of the time function $c_B(t)$, on an enhancement of the left-adjusted schedule notion and on the modelling of the problem as the search of a minimum-cost path in a directed acyclic graph.

Let $S = ((B_1, s_1), \cdots, (B_b, s_b))$ be a schedule. The allocated block (B_k, s_k) of S is said to be *strongly left-adjusted* in S if for any $t \in [f_{k-1}, s_k[$ and for any prefix B'_k of B_k, we have $c_{B'_k}(t) > c_{B'_k}(s_k)$ (with by convention $f_0 = 0$). By extension, S is said to be *strongly left-adjusted* if each of its allocated blocks is strongly left-adjusted. A prefix B'_k of B_k is said to be *left-movable* in S if $\alpha_{B'_k} < s_k$.

The following property shows that there is one optimal schedule that is strongly left-adjusted.

Property 10 *The strongly left-adjusted schedules are dominant*

Proof. — Let $S = ((B_1, s_1), \cdots, (B_b, s_b))$ be an optimal and not strongly left-adjusted schedule. Let (B_k, s_k) be *the first* non strongly left-adjusted allocated block (B_k, s_k). Let B_k^* be the smallest left-movable prefix of B_k. Notice that B_k^* is not the empty prefix since (B_k, s_k) is not strongly left-adjusted. We then define the schedule S' as follows:

First case: $\alpha_{B_k^*} \leq f_{k-1}$:

$S' = LEFTSHIFT\&MERGE(S, B_k^*, s_k)$. From the definition of B_k^* and since (B_{k-1}, s_{k-1}) is strongly left-adjusted, we derive that the allocated block $(B_{k-1}B_k^*, s_{k-1})$ is strongly left-adjusted.

Second case: $\alpha_{B_k^*} > f_{k-1}$.

$S' = LEFTSHIFT(S, B_k^*, s_k, \alpha_{B_k^*})$. From the definition of B_k^*, note that in this case the allocated block $(B_k^*, \alpha_{B_k^*})$ of S' is strongly left-adjusted.

Whatever the case, S' is still an optimal schedule and the index of the last task of the last strongly left-adjusted allocated block is stictly larger in S' than in S. So, iterating the process (at most n times) as long as the current schedule is not strongly left-adjusted yields an optimal strongly left-adjusted schedule. ∎

The block $B = (T_i, \cdots, T_j)$ is said to be *left-indivisible* if for any proper prefix B' of B we have $\alpha_{B'} \geq \alpha_B$. Similarly, B is said to be *right-indivisible* if for any proper suffix B" of B we have $\alpha_{B"} \leq \alpha_B + p(B) - p(B")$. The following property gives a strong necessary condition on the starting times of the allocated blocks of an optimal and strongly left-adjusted schedule.

Theorem 4 *Let $S = ((B_1, s_1), \cdots, (B_p, s_p))$ be an optimal and strongly left-ajusted schedule. For any $k \in \{1, \cdots, p\}$, if $s_k > 0$ then the block B_k is right and left indivisible and $s_k = \alpha_{B_k}$. If $s_1 = 0$ then B_1 is right indivisible.*

Proof. — Let S be an optimal and strongly left-adjusted schedule and let (B_k, s_k) be an allocated block of S such that $s_k > 0$. If $s_k < \alpha_{B_k}$, by right-shifting (B_k, s_k) a sufficiently small amount of time $\epsilon > 0$ we get from Property 1 a feasible schedule whose cost is strictly smaller than the cost of S, what contradicts the optimality of S. If $s_k > \alpha_{B_k}$, by left-shifting (B_k, s_k) a sufficiently small amount of time $\epsilon > 0$ we get from Property 1 either a schedule with a strictly smaller cost, what contradicts the optimality of S, or a schedule with the same cost as S, what contradicts the (strongly) left-adjusted assumption on S. We thus have $s_k = \alpha_{B_k}$ for any $k \in \{1, \cdots, p\}$ such that $s_k > 0$.
Assume that $\alpha_{B_k} > 0$ and that there is a proper prefix B'_k of B_k such that $\alpha_{B'_k} < \alpha_{B_k}$. There exists a sufficiently small $\epsilon > 0$ such that the schedule $S' = LEFTSHIFT(S, B'_k, \alpha_{B_k}, \alpha_{B_k} - \epsilon)$ is feasible. From Property 1 we then get that $c(S') \leq c(S)$, what means that S is not strongly left-adjusted.
Assume that $\alpha_{B_k} > 0$ and that there exists a suffix $B"_k$ of B_k such that $\alpha_{B"_k} > \alpha_{B_k} + p(B'_k)$. There exists a sufficiently small $\epsilon > 0$ such that the schedule $S" = RIGHTSHIFT(S, B"_k, u_k, u_k + \epsilon)$, where $u_k = \alpha_{B_k} + p(B_k) - p(B"_k)$, is feasible. From Property 1 we get that $c(S") < c(S)$, what contradicts the optimality of S.
Assume that $s_1 = 0$ and that there exists a suffix $B"_1$ of B_1 such that $\alpha_{B"_1} > \alpha_{B_1} + p(B'_1)$. There exists a sufficiently small $\epsilon > 0$ such that the schedule $S" = RIGHTSHIFT(S, B"_1, u_1, u_1 + \epsilon)$, where $u_1 = p(B_1) - p(B"_1)$, is feasible. From Property 1 we get that $c(S") < c(S)$, what contradicts the optimality of S. ∎

The necessary condition provided by Theorem 4 leads us to define the following valued directed graph called IBG (for indivisible-block graph):
The vertices of IBG are:

1. the block $B_{i,j} = (T_i, \cdots, T_j)$ if $1 \leq i \leq j \leq n$ right and if $B_{i,j}$ is a right and left-indivisible block,

2. the block $\hat{B}_{1,i}$ if $i \in \{1, \cdots, n\}$ and if $\hat{B}_{1,i}$ is a right-indivisible block ($\hat{B}_{1,i}$ corresponds to the allocated block $(B_{1,i}, 0)$),

3. a source node σ and a sink node π.

The valued arcs of IBG are:

1. for any $i \in \{1, \cdots, n\}$, the arc $(\sigma, B_{1,i})$ valued by $\gamma_{B_{1,i}}$;
2. for any $i \in \{1, \cdots, n\}$, the arc $(\sigma, \hat{B}_{1,i})$ valued by $c_{B_{1,i}}(0)$;
3. for any $i \in \{1, \cdots, n\}$, the arc $(B_{i,n}, \pi)$ valued by 0;
4. for each pair of nodes ($B_{i,j}$ and $B_{j+1,k}$) such that $\alpha_{B_{j+1,k}} - \alpha_{B_{i,j}} \geq p(B_{i,j})$ the arc $(B_{i,j}, B_{j+1,k})$ valued by $\gamma_{B_{j+1,k}}$;
5. for each pair of nodes $\hat{B}_{1,j}$ and $B_{j+1,k}$ such that $\alpha_{B_{j+1,k}} \geq p(B_{1,j})$, the arc $(\hat{B}_{1,j}, B_{j+1,k})$ valued by $\gamma_{B_{j+1,k}}$.

A path from σ to π in IBG corresponds to a schedule matching the assumptions of Theorem 4 and conversely every schedule matching these conditions corresponds to a path from σ to π in IBG. Moreover the cost of the path and the cost of the associated schedule are the same. We thus get the following property:

Property 11 *An optimal and strongly left-adjusted schedule corresponds to a minimum-cost path from σ to π in IBG.*

We propose the following two-step algorithm to compute a minimum-cost schedule: the first step builds IBG from the problem instance while the second step computes a minimum-cost path from σ to π in IBG. Note that since IBG is acyclic, the Bellman's algorithm may be used in the second step.

Worst-case complexity

The number of vertices of IBG is clearly $O(n^2)$. Since each block $B_{i,j}$ has $n - j$ immediate successors and since for fixed j, there are $j - 1$ blocks $B_{i,j}$, the number of arcs of IBG is $O(n^3)$. If $B_{i,j}$ is a block with k tasks, then by using an heap to maintain the set of the early tasks of the allocated block $(B_{i,j}, t)$ (initialy $(B_{i,j}, 0)$), the pair $(\alpha_{B_{i,j}}, \gamma_{B_{i,j}})$ may be computed in $O(k \log k)$. Thus computing all these pairs takes $O(n^3 \log n)$. Moreover deciding whether the k-tasks block $B_{i,j}$ is right and left-indivisible takes $O(k)$. So computing the nodes of IBG takes $O(n^3 \log n)$. Since searching for a minimum-cost path in IBG takes $O(n^3)$, the worst-case complexity of the

algorithm is $O(n^3 \log n)$.
Notice that restricting to the indivisible blocks increases the worst-case complexity compare to a more naive algorithm with worst-case complexity $O(n^3)$ that would consider all the blocks $B_{i,j}$ such that $1 \leq i \leq j \leq n$ and $\hat{B}_{1,i}$ such that $1 \leq i \leq n$. However it appears that in practice *many blocks are divisible* so that it is really worth taking the time to search for the indivisible blocks to get a graph with a quite smaller number of nodes.

6 Conclusion

In this paper, we first have proposed an $O(n \log n)$ algorithm for the special case of assymmetric and task-independent costs. This algorithm extends a previous algorithm by Garey et al. that applies to the case of symmetric and task-independent costs without increasing its worst-case complexity. For the general case with assymetric and task-dependent costs, we have proposed an $O(n^3 \log n)$ algorithm, which is based on a strong necessary condition on the starting times of the allocated blocks of an optimal and strongly left-adjusted schedule. We now plan to study algorithms for minimizing the mean cost per iteration for infinite periodic tasks systems.

References

[1] M.R. Garey, R.E. Tarjan and G.T. Wilfong (1988). One-processor scheduling with symmetric earliness and tardiness penalties, Maths of O.R.,13, 2, 330-348.

[2] K.R. Baker and G.D. Scudder (1989). Sequencing with Earliness-Tardiness Penalties: a Review Operations Res., 38, 1, 22-36.

[3] V. Gordon, J.M. Proth and C. Chu (1998). A State-of-the-Art Survey of Due Date Assignment and Scheduling Research: Common Due Date Rapport de recherche INRIA, 3454, theme 4.

[4] V. Gordon, J.M. Proth and C. Chu (1998). A State-of-the-Art Survey of Due Date Assignment and Scheduling Research: SLK, TWK and Other Due Date Assignment Models Rapport de recherche INRIA, 3537, theme 4.

[5] J.A. Hoogeveen and S.L. Van de Velde (1996). A branch-and-Bound Algorithm for Single-Machine Earliness-Tardiness Scheduling with Idle Time, INFORMS Journal on Computing, 8, 4, 402-412.

[6] A. Federgruen and G. Mosheiov (1997). Single-Machine Scheduling Problems with General Breakdowns, Earliness and Tardiness Costs, Operations Research, 45, 1, 66-71.

[7] A. Federgruen and G. Mosheiov (1994). Greedy Heuristics for Single-Machine Scheduling Problems with General Earliness and Tardiness Costs, Operations Research Letters, 16, 199-208.

[8] N.G. Hall, W. Kubiak and S.P. Sethi(1991). Earliness-Tardiness Scheduling Problems II. Deviation of Completion Times about a Restrictive Common Due Date, Operations Research, 39, 102-110.

[9] N.G. Hall, W. Kubiak and S.P. Sethi(1991). Earliness-Tardiness Scheduling Problems II. Deviation of Completion Times about a Restrictive Common Due Date, Operations Research, 39, 102-110.