



Mixed integer formulations using natural variables for single machine scheduling around a common due date

Anne-Elisabeth Falq, Pierre Fouilhoux, Safia Kedad-Sidhoum

► To cite this version:

Anne-Elisabeth Falq, Pierre Fouilhoux, Safia Kedad-Sidhoum. Mixed integer formulations using natural variables for single machine scheduling around a common due date. 2019. hal-02074488v1

HAL Id: hal-02074488

<https://hal.science/hal-02074488v1>

Preprint submitted on 26 Mar 2019 (v1), last revised 15 Feb 2021 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Mixed integer formulations using natural variables for single machine scheduling around a common due date

Anne-Elisabeth Falq ^a, Pierre Fouilhoux ^a, Safia Kedad-Sidhoum ^b

^a Sorbonne Université, CNRS, LIP6, 4 Place Jussieu, 75005 Paris, France

^b CNAM, CEDRIC, 292 rue Saint Martin, 75141 Paris Cedex 03, France

Abstract

While almost all existing works which optimally solve just-in-time scheduling problems propose dedicated algorithmic approaches, we propose in this work mixed integer formulations. We consider a single machine scheduling problem that aims at minimizing the weighted sum of earliness tardiness penalties around a common due-date. Using natural variables, we provide one compact formulation for the unrestrictive case and, for the general case, a non-compact formulation based on non-overlapping inequalities. We show that the separation problem related to the latter formulation is solved polynomially. In this formulation, solutions are only encoded by extreme points. We establish a theoretical framework to show the validity of such a formulation using non-overlapping inequalities, which could be used for other scheduling problems. A Branch-and-Cut algorithm together with an experimental analysis are proposed to assess the practical relevance of this mixed integer programming based methods.

1 Introduction

In the most general statement, single-machine scheduling is to process a set J of tasks non-preemptively on a single machine. Each task $j \in J$ is ready for processing at time zero and has a processing time p_j , that is neither time-dependent nor sequence-dependent (w.l.o.g. we assume that $p_j \geq 1$).

A schedule can be then encoded by the vector of its completion times $(C_j)_{j \in J}$. Such an encoding allows to express a wide range of criteria, particularly the so called regular criteria, which are decreasing functions of C_j for each task j . Using these continuous variables, Queyranne [19] provided useful polyhedral tools for minimizing one of the most studied regular criteria: $\sum \omega_j C_j$. To the best of our knowledge, the scheduling literature lacks from similar results for non-regular criteria. The contribution of this work falls within this scope. Our focus is on minimizing a non-regular criterion occurring in just-in-time scheduling.

We consider a single machine scheduling problem where all tasks share a common due date d . A task $j \in J$ is early (resp. tardy) if $C_j \leq d$ (resp. $C_j > d$). Using $[x]^+$ to denote the positive part of $x \in \mathbb{R}$, the earliness (resp. tardiness) of any task $j \in J$ is given by $[d - C_j]^+$ (resp. $[C_j - d]^+$). Given unitary earliness penalties $(\alpha_j)_{j \in J}$ (resp. tardiness penalties $(\beta_j)_{j \in J}$), the problem aims at finding a schedule that minimizes the total penalty defined as:

$$f_{\alpha, \beta}(C) = \sum_{j \in J} \alpha_j [d - C_j]^+ + \beta_j [C_j - d]^+$$

When $d \geq \sum p_j$, the common due date is called **unrestrictive** since the due date does not restrict the total duration of early tasks [12]. In this case, the so-called V-shaped dominance property [12] ensures that there exists an optimal solution such that early tasks are scheduled by increasing ratio α_j/p_j while tardy tasks are scheduled by decreasing ratio β_j/p_j . In addition, according to some strong dominance properties [12], there exists an optimal schedule without idle time and with an **on-time** task, *i.e.* completing exactly at d . The problem with an unrestrictive common due date is NP-hard even if $\alpha_j = \beta_j$ for any task $j \in J$ [12]. However, if $\alpha_j = \beta_j = 1$ for any task $j \in J$, the problem is polynomial [14].

In the general case, there might be a **straddling** job, *i.e.* a task starting before d and completing after d , in all optimal schedules: the problem is shown to be NP-hard, even if $\alpha_j = \beta_j = 1$ for all $j \in J$ [13] and [11].

In addition to these fundamental results of the common due date problem, the just-in-time field scheduling benefits from a rich literature. These problems have been treated by several approaches: with heuristics (*e.g.* [5], [16]), with branch-and-bound algorithms (*e.g.* [23]), and with dynamic programming methods (*e.g.* [13], [24]). The reader can refer to the seminal surveys of [1], [16] and [15] for the early results in this field.

Furthermore, there exist several ways to encode a single machine schedule resulting in distinct formulations. Such encodings can be based on completion times, time-indexed variables, linear ordering, positional date and assignment variables [20]. Some of these encodings allow to formulate just-in-time scheduling problems as MIP. However, few solving approaches based on these formulation have been proposed for just-in-time scheduling problems [5].

We focus in this article on natural variables, similar to completion times variables. To the best of our knowledge, no linear formulation with such variables has been considered for just-in-time scheduling, in contrast with scheduling problems dealing with regular criteria. Since tasks have to be processed on a single machine, a schedule is feasible if it satisfies the non-overlapping of tasks, *i.e.* if they are executed on disjoint time slots. Providing a linear formulation of non-overlapping is an important issue to solve a single-machine scheduling problem using linear programming. Studying the polyhedron defined as the convex hull of the feasible completion times vectors provides LP or MIP formulations. The seminal works in this research line are [2] and [19]. The authors consider the problem of minimizing $\sum \omega_j C_j$. Other works consider the same problem with additional constraints: release dates (*e.g.* [7]) or precedence constraints (*e.g.* [6], [21]).

A particularity of an encoding based on such natural variables is the non connectivity of the feasible vectors set. Therefore, a vector in the convex hull of feasible vectors can correspond to an unfeasible schedule. In this context, providing a linear formulation describing this polyhedron is not sufficient. Queyranne [19] describes the convex hull of feasible completion times vectors by linear inequalities, and shows that the extreme points of this polyhedron encode feasible schedules. He deduces a formulation which can be solved by LP algorithms. This formulation is an LP with an additional constraint: the solution must be an extreme point. This constraint will be called an **extremality constraint**.

In this article, our aim is to provide MIP based methods to solve a basic problem in just-in-time scheduling. Such approaches can be easily extended to tackle more practical problems embedding this core structure. We use natural variables to handle the common due-date problem, dealing with a non-regular criterion. Using also few binary variables, we describe a polyhedron containing the convex hull of dominant vectors for the unrestrictive case, and another one for the general case. We show that, in both cases, extreme points of this polyhedron correspond to feasible schedules, and we derive two non-compact MIP formulations with an additional extremality constraint. We explain how both formulations can be solved using a branch-and-cut algorithm. We also propose a compact MIP formulation for the unrestrictive case, which is more efficient, but which cannot be adapted to the general case. Finally we provide an experimental analysis to assess the practical relevance of the proposed approaches. The analysis is based on the reference benchmark proposed in [5] and also used in [23].

This article is organized as follows. Section 2 presents basic tools to express the tasks non-overlapping. We recall Queyranne's linear inequalities for the non-overlapping [19]. We provide also two lemmas, which permit to extend the framework in which those inequalities can be used. In Sections 3,4 and 5, we provide new formulations for the unrestrictive case and the general one. In each section we first enunciate dominance properties, then we give the formulation, and finally we prove its validity. All separation algorithms for these formulations are gathered in Section 6. In Section 7 we present some experimental results and compare the different formulations.

2 Linear inequalities for non-overlapping

For a single-machine problem, a schedule must only satisfy two conditions to be **feasible**: each task must begin after time 0 and two tasks must not be executed at the same time. In the sequel, the first condition will be called **positivity** and the second one will be called **non-overlapping**. Given the processing times $p \in (\mathbb{R}_+^*)^J$, a vector $y \in \mathbb{R}^J$ encodes a feasible schedule by its completion times if and only if it satisfies the two following constraints:

$$\text{positivity} \quad \forall j \in J, \ y_j \geq p_j \quad (0)$$

$$\text{non-overlapping} \quad \forall (i, j) \in J^2, \ y_j \geq y_i + p_j \text{ or } y_i \geq y_j + p_i \quad (1)$$

The set Q will denote the set of all vectors encoding a feasible schedule by its completion times, *i.e.* all vectors satisfying constraints (0) and (1). Completion times allow an easy way to express feasibility at the expense of the non-linearity of constraints (1). However Queyranne [19] introduces linear inequalities using completion times to handle the non-overlapping. We first recall notations and results proposed by Queyranne as we will generalize them to a larger framework. To this end, we use vector y to represent more than completion times. In the next sections, y will be either the earliness or tardiness of tasks. For $S \subseteq J$ and $y \in \mathbb{R}^J$:

$$S^< = \{(i, j) \in S^2 \mid i < j\}, \quad y(S) = \sum_{i \in S} y_i, \quad p * y(S) = \sum_{i \in S} p_i y_i, \quad \text{and} \quad g_p(S) = \frac{1}{2} \left(\sum_{i \in S} p_i \right)^2 + \frac{1}{2} \sum_{i \in S} p_i^2.$$

We give some properties about the function g_p , useful for the next proofs:

$$\forall S \subseteq J, \ g_p(S) = \sum_{(i, j) \in S^<} p_i p_j + \sum_{j \in S} p_j^2 \quad (2)$$

$$\forall S \subseteq J, \ \forall i \in J \setminus S, \ g_p(S \sqcup \{i\}) = g_p(S) + p_i(p(S) + p_i) \quad (3)$$

The non-overlapping Queyranne's inequalities are defined as follows:

$$\forall S \subseteq J, \ p * y(S) \geq g_p(S) \quad (Q0)$$

We denote P^Q the polyhedron defined by inequalities (Q0). The following property establishes that these inequalities are valid for all vectors of Q , inducing $\text{conv}(Q) \subseteq P^Q$.

Property 1

If y satisfies constraints (0) and (1), **then** y satisfies inequalities (Q0).

Proof: Let $S \subseteq J$. If $S = \emptyset$, inequality (Q0) is satisfied. If $S = \{j\}$, then inequality (Q0) is $p_j y_j \geq p_j^2$, that is $y_j \geq p_j$ since $p_j > 0$. So constraints (0) ensure that the inequalities (Q0) associated to the singletons are all satisfied.

If $|S| \geq 2$, we need to exhibit an order on J . Since processing times are strictly positive, the constraints (1) ensure that $(y_j)_{j \in J}$ are distinct and so that there exists a (single) total order \prec on J such that $i \prec j \Leftrightarrow y_i < y_j$. Then constraints (1) translate into $\forall (i, j) \in J^2, \ i \prec j \Rightarrow y_j \geq y_i + p_j$. Using inequalities (0) we deduce that $y_j \geq p(I) + p_j$ for $I \subseteq J$ and $j \in J$ such that $i \prec j$ for all $i \in I$.

This allows to prove by induction on the cardinality of S that all inequalities (Q0) are satisfied. Indeed let us assume that they are satisfied for all sets of cardinality k where $k \geq 1$ and let $S \subseteq J$ with $|S| = k+1$. By setting $j = \max_{\prec} S$ and $U = S \setminus \{j\}$, then, on one hand, by induction $p * y(U) \geq g_p(U)$, and, on the other, by previous arguments $y_j \geq p(U) + p_j$.

Consequently $p * y(S) = p * y(U) + p_j y_j \geq g_p(U) + p_j(p(U) + p_j) = g_p(S)$ using (3), hence y satisfies the inequality (Q0) associated to S . \square

Some points in $\text{conv}(Q)$ correspond to unfeasible schedules due to the disjunction inherent to the problem. Figure 2 illustrates Q and P^Q for an instance with only two tasks. The two cones represent the set of feasible schedules: each corresponding to an order in the task execution. Vectors in between correspond to schedules

where the tasks are overlapping. By definition of $\text{conv}(Q)$, these vectors are in $\text{conv}(Q)$, so they cannot be cut by the non-overlapping Queyranne's inequalities. Note that there are only two extreme points and that they correspond to feasible schedules. This observation is true in general. Indeed, Queyranne shows that the extreme points of P^Q correspond to feasible schedules. This inclusion ($\text{extr}(P^Q) \subseteq Q \subseteq \text{conv}(Q)$) and the previous one ($\text{conv}(Q) \subseteq P^Q$) are sufficient to say that $\min_{x \in Q} f(x) = \min_{x \in P^Q} f(x)$ for any given linear function f , but not sufficient to conclude that P^Q is exactly $\text{conv}(Q)$. Queyranne shows this equality using a geometrical argument, that is the equality of the two recession cones. The following theorem sums up these results.

Theorem 2 (Queyranne [19])

- (i) $\text{extr}(P^Q) \subseteq Q$
- (ii) $P^Q = \text{conv}(Q)$

Moreover, Queyranne shows that each extreme point of P^Q encodes a **left-tight** schedule, *i.e.* a feasible schedule without idle time starting at time zero. Conversely each left-tight schedule is encoded by an extreme point of P^Q since, according to the Smith rule [22], it is the only point in Q (and then in $\text{conv}(Q) = P^Q$) minimizing $\omega * C(J)$ for $\omega \in (\mathbb{R}_+)^J$ such that the tasks are scheduled by strictly decreasing ratio ω_j/p_j .

We now provide two lemmas which will be the key for showing the validity of our formulations. The first one gives also a new proof of Theorem 2(i). In this lemma, we explain how a vector of P^Q can be slightly disrupted in two directions without leaving P^Q if an overlapping is observed in the schedule it encodes. Figure 1 illustrates the two ways of disrupting the overlapping tasks so that the corresponding vectors stay in P^Q .

Lemma 3

Let us assume that y satisfies inequalities (Q0).

If there exists $(i, j) \in J^2$ with $i \neq j$ such that $y_i \leq y_j < y_i + p_j$,

then there exists $\varepsilon \in \mathbb{R}_+^*$ such that $y^{+-} = y + \frac{\varepsilon}{p_i} \mathbb{1}_i - \frac{\varepsilon}{p_j} \mathbb{1}_j$ and $y^{-+} = y - \frac{\varepsilon}{p_i} \mathbb{1}_i + \frac{\varepsilon}{p_j} \mathbb{1}_j$ also satisfy (Q0).

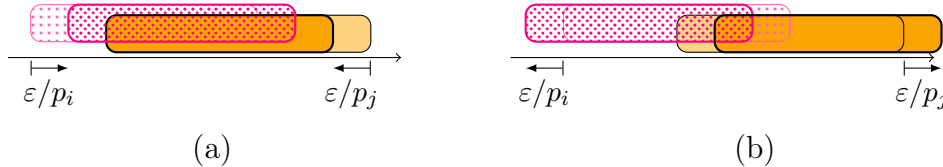


Figure 1: Illustration of the schedules disruption between y and y^{+-} (a) (resp. y^{-+} (b))

Proof: Let $\varepsilon = \min(m_1, m_2)$ where $m_1 = \min \{p * y(S) - g_p(S) \mid S \subseteq J, i \notin S, j \in S\}$

and $m_2 = \min \{p * y(S) - g_p(S) \mid S \subseteq J, i \in S, j \notin S\}$.

Since y satisfies inequalities (Q0), $m_1 \geq 0$ and $m_2 \geq 0$, thus $\varepsilon \geq 0$.

Let $S \subseteq J$. We first check that vector y^{+-} defined by ε satisfies inequality (Q0) associated to S .

If $i \notin S$ and $j \notin S$ then $p * y^{+-}(S) = p * y(S) \geq g_p(S)$.

If $i \in S$ and $j \in S$ then $p * y^{+-}(S) = p * y(S) + p_i \frac{\varepsilon}{p_i} - p_j \frac{\varepsilon}{p_j} = p * y(S) \geq g_p(S)$.

If $i \notin S$ and $j \in S$ then $p * y^{+-}(S) = p * y(S) - p_j \frac{\varepsilon}{p_j} \geq g_p(S)$ since $\varepsilon \leq m_1$.

If $i \in S$ and $j \notin S$ then $p * y^{+-}(S) = p * y(S) + p_i \frac{\varepsilon}{p_i} \geq p * y(S) \geq g_p(S)$.

In each case $p * y^{+-}(S) \geq g_p(S)$, then y^{+-} satisfies (Q0). Similarly we can check that y^{-+} satisfies (Q0) using that $\varepsilon \leq m_2$. Finally, we have to check that $\varepsilon > 0$. For this purpose we use the two next claims.

Claim

Let $(i, j) \in J^2$. If $y_i \leq y_j$, then $\forall S \subseteq J, i \notin S, j \in S \Rightarrow p * y(S) > g_p(S)$.

Proof: Let us assume on the contrary that there exists $S \subseteq J$ such that $i \notin S, j \in S$ and $p * y(S) = g_p(S)$.

Setting $U = S \setminus \{j\}$, we have $p * y(S) = p * y(U) + p_j y_j$ and $g_p(S) = g_p(U) + p_j p(S)$ by (3). Since we assume that these two terms are equal, and since $p * y(U) \geq g_p(U)$ from inequalities (Q0), we

deduce that $p_j y_j \leq p_j p(S)$ and even $y_j \leq p(S)$ since $p_j > 0$.

Moreover $p*y(S \sqcup \{i\}) = p*y(S) + p_i y_i = g_p(S) + p_i y_i \leq g_p(S) + p_i y_j$ by assumption.

Using these two inequalities, we get $p*y(S \sqcup \{i\}) \leq g_p(S) + p_i p(S) < g_p(S) + p_i [p(S) + p_i]$ since $p_i > 0$. Furthermore, $g_p(S) + p_i [p(S) + p_i] = g_p(S \sqcup \{i\})$ from (3) and $g_p(S \sqcup \{i\}) \leq p*y(S \sqcup \{i\})$ from inequality (Q0). We finally get $p*y(S \sqcup \{i\}) < p*y(S \sqcup \{i\})$, a contradiction. ■

This first claim ensures that $m_1 > 0$.

Claim

Let $(i, j) \in J^2$. **If** $y_j < y_i + p_j$, **then** $\forall S \subseteq J, i \in S, j \notin S \Rightarrow p*y(S) > g_p(S)$

Proof: Let us assume on the contrary that there exists $S \subseteq J$ such that $i \in S, j \notin S$ and $p*y(S) = g_p(S)$.

Like in the previous proof we can show that $y_i \leq p(S)$.

Moreover $p*y(S \sqcup \{j\}) = p*y(S) + p_j y_j = g_p(S) + p_j y_j < g_p(S) + p_j [y_i + p_j]$ by assumption.

Using these two inequalities, we can write $p*y(S \sqcup \{j\}) < g_p(S) + p_j [p(S) + p_j]$ since $p_j > 0$.

Furthermore, $g_p(S) + p_j [p(S) + p_j] = g_p(S \sqcup \{j\})$ from (3) and $g_p(S \sqcup \{j\}) \leq p*y(S \sqcup \{j\})$ from inequality (Q0). We finally get $p*y(S \sqcup \{j\}) < p*y(S \sqcup \{j\})$, a contradiction. ■

This second claim ensures that $m_2 > 0$, we can deduce that $\varepsilon > 0$. □

To obtain an alternative proof of Theorem 2(i), Lemma 3 can be reformulated as follows. If C is a vector of P^Q that gives the completion times of a schedule with an overlapping, then C is the middle of two other vectors of P^Q , C^{+-} and C^{-+} . That implies that C is not an extreme point of P^Q . By contraposition, we deduce that an extreme point of P^Q encodes a schedule without overlapping, and since inequalities (Q0) associated to singletons ensure the positivity, an extreme point of P^Q encodes a feasible schedule, *i.e.* $\text{extr}(P^Q) \subseteq Q$.

This way of proving that the extreme points correspond to feasible schedule can be adapted to more complex polyhedron, that are polyhedra defined by inequalities (Q0) and additional inequalities. Indeed, it suffices then to check that the two vectors C^{+-} and C^{-+} satisfy also these additional inequalities. However, for some extreme points, the two vectors introduced by Lemma 3 may not satisfy the additional inequalities. For example, if the completion times of the tasks are limited by a constant M (with $M \geq p(J)$), the additional inequalities are the following:

$$\forall j \in J, C_j \leq M \tag{4}$$

Note that these inequalities induce extreme points encoding unfeasible schedules.

On Figure 2 such an extreme point appears for a 2-tasks instance. Adding the inequalities $C_1 \leq M$ and $C_2 \leq M$ leads to the extreme point (M, M) which encodes a schedule with an overlapping. But we can see that this point will never be proposed as an optimum during the minimization of $\omega_1 C_1 + \omega_2 C_2$ if $\omega \in (\mathbb{R}_+^*)^2$. In general, the aim is to minimize a positive weighted sum of variables, so we can then only consider extreme points which minimize such a function. We introduce the following notation to denote such points for any given polyhedron P of \mathbb{R}^n :

$$\text{extr}^*(P) = \left\{ x^* \in P \mid \exists \omega \in \mathbb{R}_+^n, \{x^*\} = \underset{x \in P}{\operatorname{argmin}} \sum_{i=1}^n \omega_i x_i \right\}$$

Since an extreme point can always be written as the only minimizer of a linear function, $\text{extr}^*(P) \subseteq \text{extr}(P)$. If $P^{Q,M}$ denotes the polytope defined by inequalities (Q0) and (4), the following key lemma, shows how a vector y of $P^{Q,M}$ can be slightly disrupted in a minimization direction without leaving $P^{Q,M}$ when some y_j value is larger than $p(J)$. More precisely, if y encodes a schedule by its completion times, when a task j completing after $p(J)$ without being the last one or overlapping another task.

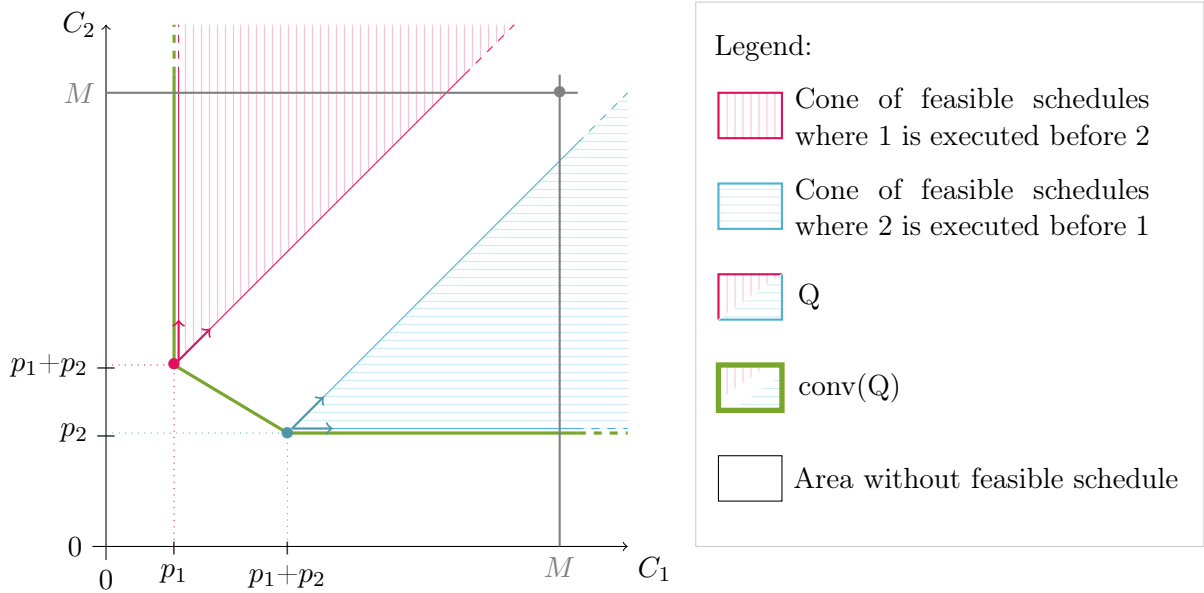


Figure 2: Q and $\text{conv}(Q)$ in the case of two tasks

Lemma 4

Let us assume that y satisfies inequalities (Q0).

If there exist $(i, j) \in J^2$ with $i \neq j$ such that $y_j < y_i + p_j$, and $y_j \geq p(J)$,
then there exists $\varepsilon \in \mathbb{R}_+^*$ such that $y - \frac{\varepsilon}{p_j} \mathbb{1}_j$ satisfies also inequalities (Q0).

Proof: Since y satisfies inequalities (Q0), setting $\varepsilon = \min\{p^*y(S) - g_p(S) \mid S \subseteq J, j \in S\}$ suffices to ensure that $y - \frac{\varepsilon}{p_j} \mathbb{1}_j$ satisfies also inequalities (Q0) and that $\varepsilon \geq 0$. So we have to show that $\varepsilon > 0$, that is to show that, for any subset $S \subseteq J$ containing j , the associated inequality (Q0) is not tight.

Let $S \subseteq J$ such that $j \in S$ and let $U = S \setminus \{j\}$. First remark the following equivalent inequalities:

$$\begin{aligned} p^*y(S) > g_p(S) &\Leftrightarrow p^*y(U) + p_j y_j > g_p(U) + p_j [p(U) + p_j] \\ &\Leftrightarrow p^*y(U) - g_p(U) > p_j [p(S) - y_j] \end{aligned}$$

If $S \subsetneq J$, then $p(S) < p(J) \leq y_j$, thus $p_j [p(S) - y_j] < 0$. Moreover $p^*y(U) - g_p(U) \geq 0$ since y satisfies the inequality (Q0) associated to T . We deduce that $p^*y(S) > g_p(S)$ in this case.

If $S = J$, then $p_j [p(S) - y_j] \leq 0$ since $y_j \geq p(J)$. In this case, $p_j [p(S) - y_j]$ can be equal to zero if $y_j = p(J)$, but we prove that $p^*y(U) - g_p(U) > 0$ as follows.

$$\begin{aligned} p^*y(U) - g_p(U) > 0 &\Leftrightarrow p^*y(J \setminus \{j\}) > g_p(J \setminus \{j\}) \\ &\Leftrightarrow p^*y(J \setminus \{i, j\}) + p_i y_i > g_p(J \setminus \{i, j\}) + p_i [p(J \setminus \{i, j\}) + p_i] \\ &\Leftrightarrow p^*y(J \setminus \{i, j\}) - g_p(J \setminus \{i, j\}) > p_i [p(J \setminus \{j\}) - y_i] \end{aligned}$$

By assumption $y_i > y_j - p_j \geq p(J) - p_j = p(J \setminus \{j\})$, thus $p_i [p(J \setminus \{j\}) - y_i] < 0$ and since y satisfies also the inequality (Q0) associated to $J \setminus \{i, j\}$, we have $p^*y(J \setminus \{i, j\}) - g_p(J \setminus \{i, j\}) \geq 0$. We deduce that $p^*y(U) - g_p(U) > 0$ in this case, and finally that $p^*y(S) > g_p(S)$. \square

Combining Lemma 3 and 4 we prove that a vector C in $\text{extr}^*(P^{Q,M})$ is in Q , that is it encodes a feasible schedule by its completion times. Indeed, since such a vector C satisfies inequalities (Q0), an overlapping between tasks i and j such that $C_i \leq C_j < C_i + p_j$ contradicts either the extremality of C or its minimality. If $C_j < p(J)$, we can construct C^{+-} and C^{-+} as proposed in Lemma 3 for ε set in $]0, p(J) - C_j[$, so that C^{+-} and C^{-+} satisfy inequalities (Q0) and (4). Thus, C can be written as the middle of two other vectors of $P^{Q,M}$, then it is not an extreme point. If conversely $C_j \geq p(J)$, we can construct a vector C^- as proposed in Lemma 4, so that C^- is component-wise smaller than C and satisfies inequalities (Q0). Thus, C^-

is another point of $P^{Q,M}$, which have a smaller value than C for any linear function with positive (or zero) coefficient, then C cannot be the single minimizer of such a function on $P^{Q,M}$. Moreover, using the same argument as for P^Q , we can say that all left-tight schedules are encoded by an extreme point of $P^{Q,M}$, and even by a vector of $\text{extr}^*(P^{Q,M})$.

For the common due date problem, an encoding by completion times does not lead to a linear objective function (except in the very particular case where $d=0$, since the tardiness are then equal to the completion times). Therefore, we propose in the next sections a schedule encoding together with a set of inequalities ensuring that every minimum extreme point corresponds to a feasible schedule.

3 A first formulation for the unrestrictive common due date problem

In this section, we consider the common due date problem when the due date is unrestrictive, *i.e.* $d \geq p(J)$. Before providing the formulation, we recall some well known dominance properties which allow not only to reduce the search space but also to restrict the instances set.

3.1 Dominance properties

We say that a set of solutions is **dominant** if it contains (at least) one optimal solution, and that it is **strictly dominant** if it contains all optimal solutions. In both cases, the search of an optimal solution can be limited to the dominant set.

For the common due-date scheduling problem, we define a **block** as a feasible schedule without idle time, a **d -schedule** as a feasible schedule with an on-time task, and a **d -block** as a block which is also a d -schedule. The following lemma gives dominance properties for the common due-date problem, already known for symmetric penalties [12]. These results can be extended to asymmetric penalties, using the same task shifting arguments.

Lemma 5

Let $\alpha \in (\mathbb{R}_+)^J$, $\beta \in \mathbb{R}_+^J$.

(i) In the general case, the blocks are dominant when minimizing $f_{\alpha,\beta}$.

Moreover, if $\alpha \in (\mathbb{R}_+^*)^J$ and $\beta \in (\mathbb{R}_+^*)^J$, the blocks are strictly dominant.

(ii) In the unrestrictive case, the d -schedules are dominant when minimizing $f_{\alpha,\beta}$

Thanks to these dominance properties, only blocks will be considered in the sequel, and only d -blocks in the unrestrictive case.

From Lemma 5, in the unrestrictive case we only have to consider instances with strictly positive earliness and tardiness penalties, *i.e.* with $\alpha \in (\mathbb{R}_+^*)^J$ and $\beta \in (\mathbb{R}_+^*)^J$. Indeed, if the tardiness penalty of a task $j \in J$ is zero, solving the instance obtained by removing task j provides a d -block, which is optimal for $J \setminus \{j\}$. Placing task j at the end of the d -block does not increase the cost, since j is then tardy. Thus, the obtained schedule is an optimal d -block. Conversely, if the earliness penalty of a task j is zero, placing task j at the beginning of an optimal d -block for $J \setminus \{j\}$ - which is always possible when d is unrestrictive - provides an optimal d -block. Hence, for the unrestrictive case, we will set $\alpha \in (\mathbb{R}_+^*)^J$ and $\beta \in (\mathbb{R}_+^*)^J$.

3.2 A natural formulation for the unrestrictive case

• A linear objective function using e and t variables

Since earliness and tardiness are not linear with respect to completion times, the objective function $f_{\alpha,\beta}$ is not linear. Therefore, we propose an encoding by earliness and tardiness of each task, by introducing the corresponding variables: $(e_j)_{j \in J}$ for the earliness of the tasks, and $(t_j)_{j \in J}$ for their tardiness. In this way, the total penalty of a schedule encoded by vector (e, t) is:

$$g_{\alpha,\beta}(e, t) = \sum_{j \in J} \alpha_j e_j + \beta_j t_j$$

which is linear. If C encodes a schedule by its completion times, the encoding by earliness and tardiness of this schedule is given by:

$$\theta(C) = \left(([d - C_j]^+)_{j \in J}, ([C_j - d]^+)_{j \in J} \right)$$

Using function θ , we can express the link between $f_{\alpha,\beta}$ and $g_{\alpha,\beta}$:

$$f_{\alpha,\beta} = g_{\alpha,\beta} \circ \theta$$

However, we have to study which vectors (e, t) encode feasible schedules. Note that there is no need to consider α and β parameters to describe this vectors set, since they only give the optimization direction.

- *Consistency between e and t using δ variables*

For a vector (e, t) in $(\mathbb{R}_+)^J \times \mathbb{R}^J$, there exists C in \mathbb{R}^J such that $\theta(C) = (e, t)$ if and only if $\forall j \in J$, $(e_j \geq 0$ and $t_j = 0)$ or $(e_j = 0$ and $t_j \geq 0)$. Such an (e, t) vector will be said **consistent**. This condition reflects that a task is either early, then its tardiness is necessarily zero, or tardy, then its earliness is necessarily zero. In order to ensure this consistency, we introduce boolean variables $(\delta_j)_{j \in J}$ representing for each task j if it is early or not, and we add the following inequalities:

$$\forall j \in J, \quad e_j \geq 0 \quad (5)$$

$$\forall j \in J, \quad e_j \leq \delta_j (p(J) - p_j) \quad (6)$$

$$\forall j \in J, \quad t_j \geq 0 \quad (7)$$

$$\forall j \in J, \quad t_j \leq (1 - \delta_j) p(J) \quad (8)$$

Inequalities (5) and (6) force e_j to be zero when $\delta_j = 0$. Since we only consider d -blocks, $p(J) - p_j$ is an upper bound of the earliness of task j . Thus, inequality (6) does not restrict e_j when $\delta_j = 1$. Note that in the unrestrictive case, $p(J) - p_j$ is tighter than $d - p_j$. Similarly, inequalities (7) and (8) force t_j to be zero when $\delta_j = 1$, without restricting t_j when $\delta_j = 0$, since $p(J)$ is an upper bound of the tardiness in a d -block. Consequently, we have the following:

Lemma 6

Let $(e, t, \delta) \in \mathbb{R}^J \times \mathbb{R}^J \times \{0, 1\}^J$.

If e, t, δ satisfy inequalities (5-8), **then** (e, t) is consistent and $C = (d - e_j + t_j)_{j \in J}$ satisfies $\theta(C) = (e, t)$.

For a consistent (e, t) vector, we define:

$$\theta^{-1}(e, t) = (d - e_j + t_j)_{j \in J}$$

Besides, inequalities (5-8) ensure the positivity of the encoded schedule. Indeed, for any j in J , inequalities (6) and (7) ensure that $d - e_j + t_j \geq d - e_j \geq d - p(J) + p_j$. Since d is unrestrictive, we deduce that $d - e_j + t_j \geq p_j$. Hence, we obtain:

Lemma 7

Let $(e, t, \delta) \in \mathbb{R}^J \times \mathbb{R}^J \times \{0, 1\}^J$. **If** e, t, δ satisfy (5-8), **then** $\theta^{-1}(e, t)$ satisfies (0).

- *Handling the non-overlapping*

To ensure the non-overlapping, it suffices that early tasks are fully processed before d and that they do not overlap each other, and that tardy tasks are fully processed after d and that they do not overlap each other either. Note that for a d -schedule, the non-overlapping reduces to these two constraints related to early and tardy tasks respectively.

In order to use the partition between early and tardy tasks induced by the completion times C , we introduce the following notations:

$$E(C) = \{j \in J \mid C_j \leq d\} \quad \text{and} \quad T(C) = \{j \in J \mid C_j > d\}$$

For a tardy task, the tardiness plays the same role regarding to d as the completion time with respect to 0. Then ensuring that the tardy tasks are fully processed before d (resp. they do not overlap each other) is equivalent to the positivity constraint for tardy tasks (resp. to the non-overlapping constraint for tardy tasks). As shown on Figure 3, the situation is almost the same for an early task, the only difference is that it is not the earliness e_j which plays the same role as the completion time, but $e_j + p_j$. Using $x_{/S}$ to denote $(x_j)_{j \in S}$ for any subset S of J and for any vector x in \mathbb{R}^J , the following lemma sums up these observations.

Lemma 8

Let $C \in \mathbb{R}^J$ and set $(e, t) = \theta(C)$. **If** there exists $j \in J$ such that $C_j = d$,

then C satisfies (1) $\Leftrightarrow \begin{cases} (e+p)_{/E(C)} & \text{satisfies (0) and (1)} \\ t_{/T(C)} & \text{satisfies (0) and (1)} \end{cases}$

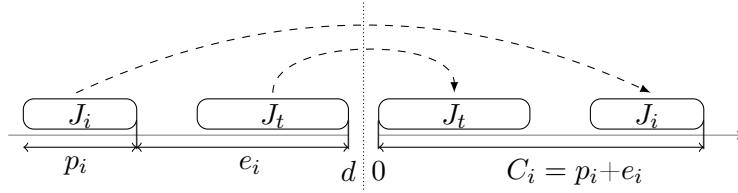


Figure 3: Illustration of the role of $p_i + e_i$ for an early task i

In the formulation, we use δ to describe the partition between early and tardy tasks, that is:

$$E(\delta) = \{j \in J \mid \delta_j = 1\} \quad \text{and} \quad T(\delta) = \{j \in J \mid \delta_j = 0\}.$$

According to Section 2, we want to apply Queyranne's inequalities (Q0) to the vectors $(e+p)_{/E(\delta)}$ and $t_{/T(\delta)}$ respectively, so that they satisfy (0) and (1). Therefore, we consider the following inequalities:

$$\forall S \subseteq J, \quad p * (e+p)(S \cap E(\delta)) \geq g_p(S) \quad (9)$$

$$\forall S \subseteq J, \quad p * t(S \cap T(\delta)) \geq g_p(S) \quad (10)$$

These inequalities are not linear as $E(\delta)$ and $T(\delta)$ depend on δ variables. Replacing $S \cap E(\delta)$ (resp. $S \cap T(\delta)$) by S raises non valid inequalities. Indeed, inequality (10) for $S = \{i, j\}$ where $i \in E$, would become $p_j t_j \geq p_i^2 + p_j^2 + p_i p_j$ since $t_i = 0$ by (7) and (8). This implies that $t_j > p_j$, which is not valid for all the feasible schedules.

To ensure that only the terms corresponding to early (resp. tardy) tasks are involved in (9) (resp. in (10)), we multiply each term of index j in S in the inequality by δ_j (resp. by $(1 - \delta_j)$). We obtain the following quadratic inequalities:

$$\forall S \subseteq J, \quad \sum_{j \in S} p_j e_j \geq \sum_{(i,j) \in S^<} p_i p_j \delta_i \delta_j \quad (11)$$

$$\forall S \subseteq J, \quad \sum_{j \in S} p_j t_j (1 - \delta_j) \geq \sum_{(i,j) \in S^<} p_i p_j (1 - \delta_i)(1 - \delta_j) + \sum_{j \in S} p_j^2 (1 - \delta_j) \quad (12)$$

• *Linearization of the quadratic terms using x variables*

In order to remove the quadratic terms, we introduce a new variable $x_{i,j}$ representing whether δ_i is different than δ_j for each (i, j) in $J^<$. Since the quadratic terms are the products of boolean variables, the following inequalities ensure their consistency with respect to δ :

$$\forall (i, j) \in J^<, \quad x_{i,j} \geq \delta_i - \delta_j \quad (13)$$

$$\forall (i, j) \in J^<, \quad x_{i,j} \geq \delta_j - \delta_i \quad (14)$$

$$\forall (i, j) \in J^<, \quad x_{i,j} \leq \delta_i + \delta_j \quad (15)$$

$$\forall (i, j) \in J^<, \quad x_{i,j} \leq 2 - (\delta_i + \delta_j) \quad (16)$$

The following lemma provides the correspondence between quadratic and linear terms.

Lemma 9

If $\delta \in \{0, 1\}^J$ **then** for all $(i, j) \in J^<$:

- (i) δ and x satisfy the inequalities (13-16) associated to $(i, j) \Leftrightarrow x_{i,j} = \begin{cases} 0 & \text{if } \delta_i = \delta_j \\ 1 & \text{if } \delta_i \neq \delta_j \end{cases}$
- (ii) In case (i) holds, then $\delta_i \delta_j = \frac{\delta_i + \delta_j - x_{i,j}}{2}$ and $(1 - \delta_i)(1 - \delta_j) = \frac{2 - (\delta_i + \delta_j) - x_{i,j}}{2}$

The proof can be easily done by considering the two cases $\delta_i = \delta_j$ and $\delta_i \neq \delta_j$.

- *Non-overlapping inequalities*

Using Lemma 9(ii), we obtain the following inequalities:

$$\forall S \subseteq J, \sum_{i \in S} p_i e_i \geq \sum_{(i,j) \in S^<} p_i p_j \frac{\delta_i + \delta_j - x_{i,j}}{2} \quad (\text{Q1})$$

$$\forall S \subseteq J, \sum_{i \in S} p_i t_i \geq \sum_{(i,j) \in S^<} p_i p_j \frac{2 - (\delta_i + \delta_j) - x_{i,j}}{2} + \sum_{i \in S} p_i^2 (1 - \delta_i) \quad (\text{Q2})$$

The following lemma summarizes the relationship between the inequalities (Q1), (Q2) and (Q0).

Lemma 10

Let $(\delta, x) \in \{0, 1\}^J \times \mathbb{R}^{J^<}$ such that inequalities (13-16) are satisfied.

(i) **If** $e \in \mathbb{R}^J$ satisfies inequalities (5) and (6) for all $j \in E(\delta)$, **then**:

e, δ, x satisfy inequalities (Q1) for all $S \subseteq J$ iff $(e+p)_{/E(\delta)}$ satisfies inequalities (Q0)

(ii) **If** $t \in \mathbb{R}^J$ satisfies inequalities (7) and (8) for all $j \in T(\delta)$, **then**:

t, δ, x satisfy inequalities (Q2) for all $S \subseteq J$ iff $t_{/T(\delta)}$ satisfies inequalities (Q0)

The following lemma allows to make the bridge between $(e+p)_{/E(C)}$ from Lemma 8 and $(e+p)_{/E(\delta)}$ from Lemma 10 (resp. between $t_{/T(C)}$ and $t_{/T(\delta)}$).

Lemma 11

Let $(e, t, \delta) \in \mathbb{R}^J \times \mathbb{R}^J \times \{0, 1\}^J$.

If e, t, δ satisfy (5-8) and (Q2) **then** $E(\delta) = E(\theta^{-1}(e, t))$ and $T(\delta) = T(\theta^{-1}(e, t))$.

Proof: Let $C = \theta^{-1}(e, t)$. If $j \in T(C)$, then $C_j > d$ by definition. That is $t_j > e_j$ since $C_j = d - e_j + t_j$. It implies that $t_j > 0$, using inequality (5), and using inequality (8), we deduce that $\delta_j \neq 1$, and thus $\delta_j = 0$ since δ_j is integer. That proves $T(C) \subseteq T(\delta)$. Conversely, if $j \in T(\delta)$, inequalities (5) and (6) ensure that $e_j = 0$, since $\delta_j = 0$ by definition. Thus, $C_j = d + t_j$. Since $t_j > p_j > 0$ from inequality (Q2) for $S = \{j\}$, we deduce that $C_j > d$, that proves $T(\delta) \subseteq T(C)$.

Similarly, we can prove the equality for the early tasks (without using (Q1)). □

- *Formulation (F1)*

Let us define the polyhedron:

$$P^1 = \{ (e, t, \delta, x) \in \mathbb{R}^J \times \mathbb{R}^J \times [0, 1]^J \times \mathbb{R}^{J^<} \mid (5-8), (13-16), (Q1) \text{ and } (Q2) \text{ are satisfied} \}$$

Note that this polyhedron does not depend on either α , β , or even d , but is defined only from p . Moreover, this polyhedron is defined by an exponential number of inequalities, inducing the use of a separation algorithm, this subject will be the purpose of Section 6.

Since δ are boolean variables, we are only interested in vectors for which δ is integer, that are integer points. Therefore, we introduce the operator int_δ , which only keeps the integer points of a set. For V included in $\mathbb{R}^J \times \mathbb{R}^J \times \mathbb{R}^J \times \mathbb{R}^{J^<}$,

$$\text{int}_\delta(V) = \{ (e, t, \delta, x) \in V \mid \delta \in \{0, 1\}^J \}.$$

However, the formulation is not a classical MIP formulation, since some integer points do not encode feasible schedules. The same observation holds for $P^{Q,M}$, as discussed in Section 2 (apart the integrity constraints on δ). Therefore, we need to add an extremality condition (and consider the minimality condition) to ensure the feasibility. Finally, our formulation for the unrestrictive common due date problem defined by the unit penalties (α, β) is:

$$(F1) \quad \begin{array}{ll} \min & g_{\alpha, \beta}(e, t) \\ \text{s.t.} & (e, t, \delta, x) \in \text{int}_\delta(\text{extr}(P^1)) \end{array}$$

3.3 Validity of Formulation (F1)

The following theorem establishes that a feasible schedule, under some assumptions, is encoded by an integer point of P^1 . In particular a d -block is encoded by an integer point of P^1 .

Theorem 12

If vector C gives the completion times of a feasible schedule without straddling task such that tasks are processed between $d-p(J)$ and $d+p(J)$, i.e. $\forall j \in J, d-p(J) \leq C_j - p_j$ and $C_j \leq d+p(J)$
then there exists $X = (e, t, \delta, x) \in \text{int}_\delta(P^1)$, such that $\theta(C) = (e, t)$.

Proof: From C , let us set: $(e, t) = \theta(C)$, $\delta = \mathbb{1}_{E(C)}$, $x = (\mathbb{1}_{\delta_i \neq \delta_j})_{(i,j) \in J^<}$ and $X = (e, t, \delta, x)$.

Note that the definition of δ ensures that $\delta \in \{0, 1\}^J \subseteq [0, 1]^J$, and that $E(\delta) = E(C)$ (resp. $T(\delta) = T(C)$), which allows the notation E (resp T) for sake of brevity. Inequalities (5) and (7), as well as (6) for j in T and (8) for j in E , are automatically satisfied by construction of e, t and δ . The assumption that $\forall j \in J, d-p(J) \leq C_j - p_j$ (resp. $C_j \leq d+p(J)$) ensures that inequalities (6) for j in E (resp. inequalities (8) for j in T) are satisfied.

Using Lemma 9(i), x and δ satisfy inequalities (13-16).

Since C encodes a feasible schedule, C satisfies (0) and (1). Using Lemma 8, $(e+p)_{/E}$ (resp. $t_{/T}$) satisfies (0) and (1). Applying Property 1 to these two vectors, we deduce that they satisfy (Q0), and using Lemma 10, that e, δ, x satisfy (Q1) and t, δ, x satisfy (Q2). Thus, X belongs to P^1 , and even in $\text{int}_\delta(P^1)$ since $\delta \in \{0, 1\}^J$. \square

The following theorem establishes that an optimal solution of formulation (F1) is a solution for the unrestrictive common due date problem.

Theorem 13

Let $X^* = (e, t, \delta, x) \in \text{int}_\delta(P^1)$.
If $X^* \in \text{extr}(P^1)$ and (e, t) minimizes $g_{\alpha, \beta}$ **then** X^* encodes a d -block.

Proof: The first step is to show that X^* encodes a feasible schedule.

From Lemma 6, X^* is consistent and we can set $C^* = \theta^{-1}(e, t)$. Then X^* encodes a schedule defined by the completion times C^* . This schedule will be denoted \mathcal{S}^* . Proving that \mathcal{S}^* is feasible consists then in showing that C^* satisfies (0) and (1). From Lemma 7, C^* satisfies (0). From Lemma 11, $E(\delta) = E(C^*)$ (resp. $T(\delta) = T(C^*)$), which allows the notation E (resp. T) for sake of brevity. Using Lemma 8, to show that C^* satisfies (1), it remains to show that $(e+p)_{/E}$ (resp. $t_{/T}$) satisfies (0) and (1).

From Lemma 10, we know that $(e+p)_{/E}$ (resp. $t_{/T}$) satisfies inequalities (Q0).

On one hand, using these inequalities for the singletons, ensures that $(e+p)_{/E}$ (resp. $t_{/T}$) satisfies (0). We deduce that no straddling task occurs in \mathcal{S}^* .

On the other hand, inequalities (Q0) will allow us to show that $(e+p)_{/E}$ (resp. $t_{/T}$) satisfies (1) in the same way that we have shown that a vector in $\text{extr}^*(P^{Q,M})$ encodes a schedule without overlapping in Section 2.

Let us assume that $(e+p)_{/E}$ does not satisfy (1). Then there exists $(i, j) \in E^2$ such that $e_i + p_i \leq e_j + p_j < (e_i + p_i) + p_j$. Two cases have to be considered:

\rightarrow If $e_j + p_j < p(J)$, then by Lemma 3 applied to $(e+p)_{/E}$, there exists $\varepsilon \in \mathbb{R}_+^*$ such that, setting

$$e^{+-} = e + \frac{\varepsilon}{p_i} \mathbb{1}_i - \frac{\varepsilon}{p_j} \mathbb{1}_j \quad \text{and} \quad e^{-+} = e - \frac{\varepsilon}{p_i} \mathbb{1}_i + \frac{\varepsilon}{p_j} \mathbb{1}_j,$$

both $(e^{+-} + p)_{/E}$ and $(e^{-+} + p)_{/E}$ satisfy (Q0). Using Lemma 10, both e^{+-} and e^{-+} satisfy (Q2). Since changing the value of ε for $\min(\varepsilon, p(J) - p_j - e_j)$ does not affect the satisfaction of (Q0), we can assume that $\varepsilon \leq p(J) - p_j - e_j$, while ensuring $\varepsilon > 0$. Since $e_i^{+-} = e_i + \frac{\varepsilon}{p_i} \leq e_i + \varepsilon$, using this latter assumption and that $e_j + p_j \geq e_i + p_i$, we obtain that $e_i^{+-} \leq p(J) - p_i$. For k in $J \setminus \{i\}$, $e_k^{+-} \leq e_k$, and since e satisfies (6), we deduce that $e_k^{+-} \leq p(J) - p_k$.

Thus e^{+-} satisfies inequalities (6). Besides, since $(e^{+-} + p)_{/E}$ satisfies inequalities (Q1) for the singletons, $e_k^{+-} + p_k \geq p_k$ for all k in E . Since $e_k^{+-} = e_k$ for all k in T and e satisfies (5), we deduce that e^{+-}

satisfies inequalities (5). Similarly e^{-+} satisfies inequalities (5) and (6). Finally, $X^{+-} = (e^{+-}, t, \delta, x)$ and $X^{-+} = (e^{-+}, t, \delta, x)$, are two points of P^1 whose middle point is X^* . A contradiction, since X^* is extreme.

→ If $e_j + p_j \geq p(J)$, then $e_j + p_j \geq p(E)$, and by Lemma 4 applied to $(e+p)_{/E}$, there exists $\varepsilon \in \mathbb{R}_+^*$ such that, setting $e^- = e - \frac{\varepsilon}{p_j} \mathbb{1}_j$, $(e^- + p)_{/E}$ satisfies (Q0). Using Lemma 10, e^- satisfies (Q2). Since e^- is component wise smaller than e , then e^- also satisfies inequalities (6). Besides, the inequality (Q0) for the singleton $\{j\}$ ensures that $e_j^- \geq 0$, thus e^- satisfies inequalities (5). Finally, setting $X^- = (e^-, t, \delta, x)$, we exhibit a point of P^1 , which has a smaller value than X^* according to $g_{\alpha, \beta}$. A contradiction, since (e, t) minimizes $g_{\alpha, \beta}$. Consequently $(e+p)_{/E}$ satisfies (1).

In the same way, we can prove that $t_{/T}$ satisfies (1). We deduce that \mathcal{S}^* is a feasible schedule.

The second step consists in showing that \mathcal{S}^* is a d -block.

Since we already know that \mathcal{S}^* does not hold a straddling task, it suffices to show that it is a block with at least one early task to conclude that is a d -block.

Let us assume that \mathcal{S}^* holds an idle time or has no early task. Let $\widehat{\mathcal{S}}$ denotes the schedule obtained by tightening tasks around d to fill idle times and, if there is no early task, shifting backward all tasks such that the first one becomes on-time. By construction, $\widehat{\mathcal{S}}$ is a block and is feasible, since the due date is unrestrictive. If \widehat{C} denotes the completion times defining $\widehat{\mathcal{S}}$, then $\forall j \in J$, $d - p(J) \leq \widehat{C}_j - p_j$ and $\widehat{C}_j \leq d + p(J)$.

Then using Theorem 12, there exists $\widehat{X} = (\widehat{e}, \widehat{t}, \widehat{\delta}, \widehat{x}) \in \text{int}_\delta(P^1)$, such that $\theta(\widehat{C}) = (\widehat{e}, \widehat{t})$. Moreover, $f_{\alpha, \beta}(\widehat{C}) < f_{\alpha, \beta}(C^*)$, since the early tasks stay early but with a smaller earliness, and the tardy tasks, except the first tardy task which becomes eventually on-time, stay tardy with a smaller earliness.

Then $g_{\alpha, \beta}(\widehat{e}, \widehat{t}) = f_{\alpha, \beta}(\widehat{C}) < f_{\alpha, \beta}(C^*) = g_{\alpha, \beta}(e, t)$, which contradicts the minimality of (e, t) .

Finally, X^* encodes a d -block. □

The following theorem establishes that the unrestrictive common due date problem reduces to solving formulation (F1).

Theorem 14

Any optimal d -block, is encoded by a vector minimizing $g_{\alpha, \beta}$ on $\text{int}_\delta(\text{extr}(P^1))$.
Conversely, any vector minimizing $g_{\alpha, \beta}$ on $\text{int}_\delta(\text{extr}(P^1))$, encodes an optimal d -block.

Proof: Let us consider an optimal d -block \mathcal{S}^* . From Theorem 12, there exists a vector $X^* = (e^*, t^*, \delta^*, x^*)$ in $\text{int}_\delta(P^1)$ encoding \mathcal{S}^* . We introduce:

$$P^{\delta^*} = \{(e, t) \mid (e, t, \delta^*, x^*) \in P^1\},$$

which is the slice of P^1 according to δ^* , i.e. the set of points of P^1 satisfying $\delta = \delta^*$, and therefore $x = x^*$. To show that X^* is also an extreme point of P^1 , it suffices to prove that (e^*, t^*) is an extreme point of P^{δ^*} . Indeed, if there were $X^1 = (e^1, t^1, \delta^1, x^1)$ and $X^2 = (e^2, t^2, \delta^2, x^2)$ in P^1 such that $X^* = \frac{1}{2}(X^1 + X^2)$, δ^1 and δ^2 would necessarily be equal to δ^* since $\delta^* \in \{0, 1\}^J$, $\delta^1 \in [0, 1]^J$ and $\delta^2 \in [0, 1]^J$. By Lemma 9, we deduce that $x^1 = x^*$ (resp. $x^2 = x^*$), and thus (e^1, t^1) (resp. (e^2, t^2)) is in P^{δ^*} . Yet $(e^*, t^*) = \frac{1}{2}((e^1, t^1) + (e^2, t^2))$, and (e^*, t^*) would not be an extreme point of P^{δ^*} .

Using E and T to denote the partition of tasks given by δ^* , i.e. $E = E(\delta^*)$ and $T = T(\delta^*)$, we decompose P^{δ^*} as a Cartesian products of polyhedra:

$$P^{\delta^*} = P^{\delta^*, E} \times \{0\}^T \times P^{\delta^*, T} \times \{0\}^E \text{ where } \begin{cases} P^{\delta^*, E} = \{\tilde{e} \in \mathbb{R}^E \mid \tilde{e} + p_{/E} \text{ satisfies (Q0) and } \forall j \in E, \tilde{e}_j + p_j \leq p(J)\} \\ P^{\delta^*, T} = \{\tilde{t} \in \mathbb{R}^T \mid \tilde{t} \text{ satisfies (Q0) and } \forall j \in T, \tilde{t}_j \leq p(J)\} \end{cases}$$

Knowing that the extreme points set of a Cartesian product is exactly the Cartesian product of the extreme points sets, it remains to show that $e_{/E}^* \in \text{extr}(P^{\delta^*, E})$ and that $t_{/T}^* \in \text{extr}(P^{\delta^*, T})$. Note that $P^{\delta^*, T}$

is the polyhedron called $P^{Q,M}$ in Section 2, replacing the index set J by T and setting $M = p(J) \geq p(T)$. Similarly $P^{\delta^*,E}$ is a translation according to $-p/E$ of $P^{Q,M}$, replacing J by E and setting $M = p(J) \geq p(E)$. Then it suffices that $t_{/T}^*$ (resp. $e_{/E}^* + p/E$) encodes a left-tight schedule of tasks in T (resp. E) to ensure its extremality in $P^{\delta^*,T}$ (resp. $P^{\delta^*,E}$). Both conditions are satisfied since X^* encodes a d -block. We deduce that (e^*, t^*) belongs to $\text{extr}(P^{\delta^*})$. Thus X^* belongs to $\text{int}_\delta(\text{extr}(P^1))$.

According to Theorem 13, another point in $\text{int}_\delta(\text{extr}(P^1))$ with a smaller value according to $g_{\alpha,\beta}$ would correspond to a better schedule, yet \mathcal{S}^* is optimal. We then deduce the minimality of X^* on $\text{int}_\delta(\text{extr}(P^1))$ from the optimality of \mathcal{S}^* .

The second part of the theorem is then a direct corollary of Theorem 13. The schedule encoded by a vector X^* minimizing $g_{\alpha,\beta}$ on $\text{int}_\delta(\text{extr}(P^1))$ is a d -block, and if it was not optimal, there would exist a strictly better d -block, and then a vector in $\text{int}_\delta(\text{extr}(P^1))$ with a smaller value according to $g_{\alpha,\beta}$, a contradiction. \square

3.4 Dealing with Formulation (F1)

The aim of this section is to show that formulation (F1) can be solved by a classical branch-and-cut algorithm. Let us consider three relaxations of (F1):

$$\begin{array}{lll} \text{(F1-LP)} & \min g_{\alpha,\beta}(e, t) \\ & \text{s.t. } (e, t, \delta, x) \in P^1 & \text{(F1-extr)} \quad \min g_{\alpha,\beta}(e, t) \\ & & \text{s.t. } (e, t, \delta, x) \in \text{extr}(P^1) & \text{(F1-int)} \quad \min g_{\alpha,\beta}(e, t) \\ & & & \text{s.t. } (e, t, \delta, x) \in \text{int}_\delta(P^1) \end{array}$$

The formulation (F1-LP) is obtained by relaxing the integrity and the extremality conditions. It is a linear program defined by an exponential number of inequalities. We will explain in Section 6 that the separation problem associated to the non-overlapping inequalities defining P^1 is solvable in polynomial time. Then (F1-LP) can be solved in polynomial time using a cutting plane algorithm [10].

Using the simplex algorithm for each LP-relaxation of a cutting plane algorithm, the extremality of the solution is ensured. Then in this case, solving (F1-LP) is equivalent to solving (F1-extr).

A classical way to manage the integrity constraint is to use a branch-and-bound algorithm, and even in this case a branch-and-cut algorithm.

Using an algorithm which provides an extreme point to solve each LP-relaxation, a branch-and-bound algorithm directly computes a solution of (F1):

Property 15

Let us consider a branch-and-bound algorithm \mathcal{A} , where the LP-relaxation at each node provides an extreme point. Using \mathcal{A} to solve (F1-int) by branching on δ variables solves (F1).

Proof: By assumption, the solution provided at each node of the branch-and-bound tree is an extreme point of the polyhedron defined by the decisions previously taken, and we will prove that this solution is also an extreme point of P^1 .

Formally, if variables δ_j for $j \in J_0$ (resp. for $j \in J_1$) have been fixed to 0 (resp. to 1), the polyhedron considered is $P^1 \cap F^{J_0, J_1}$ where:

$$F^{J_0, J_1} = \{ (e, t, \delta, x) \in \mathbb{R}^J \times \mathbb{R}^J \times [0, 1]^J \times [0, 1]^{J^<} \mid \forall j \in J_0, \delta_j = 0 \text{ and } \forall j \in J_1, \delta_j = 1 \}$$

We consider an arbitrary node defined by J_0 and J_1 , and a vector $X = (e, t, \delta, x) \in \text{extr}(P^1 \cap F^{J_0, J_1})$.

By definition of $E(\delta)$ and $T(\delta)$, $X \in P^1 \cap F^{T(\delta), E(\delta)}$. Moreover, $J_1 \subseteq E(\delta)$ and $J_0 \subseteq T(\delta)$, thus we have $P^1 \cap F^{T(\delta), E(\delta)} \subseteq P^1 \cap F^{J_0, J_1}$. Recall that if $A \subseteq B$, then $\text{extr}(B) \cap A \subseteq \text{extr}(A)$, we deduce that $X \in \text{extr}(P^1 \cap F^{T(\delta), E(\delta)})$. Since $P^1 \cap F^{T(\delta), E(\delta)}$ is exactly the set denoted P^δ in the previous proof, we get $\text{extr}(P^1 \cap F^{T(\delta), E(\delta)}) \subseteq \text{extr}(P^1)$. We deduce that $X \in \text{extr}(P^1)$. \square

Property 15 stands for the formulation (F1) since the integrity constraint on δ_j can be rewritten as $\delta_j \in \text{extr}([0, 1])$, for any $j \in J$. Therefore, the integrity of δ and the extremality in P^{δ^*} induce the extremality

in P^1 . Note that the property is not valid for an arbitrary formulation. To illustrate this observation, let us consider the following formulation:

$$(F) \quad \begin{array}{ll} \max & z \\ \text{s.t.} & (y, z) \in \text{int}_y(\text{extr}(P)) \end{array} \quad \text{with } P = \{(y, z) \in \mathbb{R}_+ \times \mathbb{R}_+ \mid z \leq \frac{2}{3}y + 2, \quad z \leq -2y + 6\},$$

where int_y denotes the operator keeping only the points with an integer y . \mathcal{A} provides a solution which does not belong to $\text{extr}(P)$. Indeed, since $(\frac{3}{2}, 3)$ is the solution at the root node, the search space is divided into $P \cap]-\infty, 1] \times \mathbb{R}$ and $P \cap [2, +\infty[\times \mathbb{R}$, and the extreme points maximizing z in these polyhedron are respectively $(1, 2 + \frac{2}{3})$, and $(2, 2)$. Then the provided point is $(1, 2 + \frac{2}{3})$, with a value of $2 + \frac{2}{3}$ whereas the best value for an integer extreme point is 2, reached by $(0, 2)$.

For any formulation F , let us denote by $\text{value}(F)$ the value of any optimal solution for the optimization problem F . Using any algorithm to solve each LP-relaxation, a branch-and-bound algorithm can solve (F1-int), that gives $\text{value}(F1)$, but not directly a solution of (F1). Indeed, if $X = (e, t, \delta, x)$ denotes the provided vector, δ is 0-1 and (e, t) minimizes $g_{\alpha, \beta}$ on P^δ by construction. Then, there exists (e^*, t^*) in $\text{extr}(P^\delta)$ such that $g_{\alpha, \beta}(e^*, t^*) = g_{\alpha, \beta}(e, t)$. Since $X^* = (e^*, t^*, \delta, x) \in \text{int}_\delta(\text{extr}(P^1))$, we get :

$$g_{\alpha, \beta}(e^*, t^*) \geq \text{value}(F1) \geq \text{value}(F1\text{-int}) = g_{\alpha, \beta}(e, t)$$

In addition to this theoretical way to come down to an extreme point, and then to a feasible solution, there is a computational way to do that from the partition between early and tardy tasks defined by δ . It will be the purpose of the next section.

4 A second formulation for the unrestrictive case

The unrestrictive common due date problem is NP-hard, so the decision problem associated to (F1) is NP-hard. In contrast, (F1-extr) is solvable in polynomial time. We deduce that the hardness of the formulation (F1) is only due to the integrity constraints on δ variables. This suggests that the main difficulty of the unrestrictive common due date problem lies in choosing which tasks are early and which ones are tardy. This observation is corroborated by the following dominance property, known in the just-in-time scheduling field, which ensures that, in the unrestrictive case, once the partition between early and tardy tasks is fixed, it suffices to sort tasks to obtain an optimal schedule. A question is then: how to exploit the strength of this property in a linear way? This issue leads to a compact formulation for the unrestrictive case, presented in this section.

4.1 Dominance properties

We recall some dominance properties known for the symmetric penalties case [13], but given here in their most general statement.

Lemma 16

Let $\alpha \in (\mathbb{R}_+)^J$, $\beta \in (\mathbb{R}_+)^J$.

In the general case, the schedules where the tasks ending before or at d (resp. starting at or after d) are in order of nondecreasing α_j/p_j (resp. nonincreasing β_j/p_j) are strictly dominant when minimizing $f_{\alpha,\beta}$.

For given unit penalties α and β , a feasible schedule is said **V-shaped** if the early tasks in order of increasing α_j/p_j and the tardy ones in order of decreasing β_j/p_j . Since the tasks ending before or at d are exactly the early ones in any schedule, and the tasks starting after or at d are exactly the tardy ones in a d -schedule, we deduce from Lemma 5 and Lemma 16, that V-shaped d -blocks are dominant in the unrestrictive case.

In case of equality between two ratios α_i/p_i and α_j/p_j (resp. β_i/p_i and β_j/p_j), swapping tasks i and j does not change the total penalty of a schedule if both are early (resp. tardy). Thus in this case, there exist different optimal V-shaped d -blocks with the same partition between early and tardy tasks. To ensure there is only one way to decode a partition between early and tardy tasks into a dominant schedule, we fix *a priori* two orders on tasks : one by decreasing α_j/p_j , and one by decreasing β_j/p_j . Let ρ and σ denote two functions in $J^{\llbracket 1,n \rrbracket}$ such that:

$$\left(\frac{\alpha_{\rho(k)}}{p_{\rho(k)}} \right)_{k \in \llbracket 1,n \rrbracket} \quad \text{and} \quad \left(\frac{\beta_{\sigma(k)}}{p_{\sigma(k)}} \right)_{k \in \llbracket 1,n \rrbracket} \quad \text{are decreasing.}$$

We say that a feasible schedule is **ρ - σ -shaped** when early (resp. tardy) tasks are processed in order of decreasing $\rho(j)$ (resp. increasing $\sigma(j)$). These schedules are dominant in the unrestrictive case, and only them will be considered for the remainder of this section. Note that there is a one-to-one correspondence between the ρ - σ -shaped d -blocks and the vectors $\delta \in \{0,1\}^J$.

4.2 A compact formulation for the unrestrictive case

If the partition between early and tardy tasks of a ρ - σ -shaped d -block is given by δ , then the earliness and tardiness are given by:

$$e^\rho(\delta) = \left(\delta_j \times \sum_{k=1}^{\rho^{-1}(j)-1} p_{\rho(k)} \delta_{\rho(k)} \right)_{j \in J} \quad \text{and} \quad t^\sigma(\delta) = \left((1-\delta_j) \times \sum_{k=1}^{\sigma^{-1}(j)} p_{\sigma(k)} (1-\delta_{\sigma(k)}) \right)_{j \in J}$$

Using the same x variables as done in Section 3 to linearize these terms, we consider:

$$e^\rho(\delta, x) = \left(\sum_{k=1}^{\rho^{-1}(j)-1} p_{\rho(k)} \frac{\delta_j + \delta_{\rho(k)} - x_{j,\rho(k)}}{2} \right)_{j \in J} \quad \text{and} \quad t^\sigma(\delta, x) = \left(\sum_{k=1}^{\sigma^{-1}(j)-1} p_{\sigma(k)} \frac{2 - (\delta_j + \delta_{\sigma(k)}) - x_{j,\sigma(k)}}{2} + p_j(1-\delta_j) \right)_{j \in J}$$

where we use $x_{i,j}$ without carrying if $i < j$, that is to denote the variable $x_{\min(i,j),\max(i,j)}$. Hence the total penalty is simply expressed by:

$$h_{\alpha,\beta}^{\rho,\sigma}(\delta, x) = f_{\alpha,\beta}(e^\rho(\delta, x), t^\sigma(\delta, x)),$$

which is linear. We then consider the following polyhedron:

$$\mathbf{P}^2 = \{ (\delta, x) \in [0, 1]^J \times \mathbb{R}^{J^<} \mid (13-16) \text{ are satisfied} \}.$$

By definition of $e^\rho(\delta, x)$ and $t^\sigma(\delta, x)$, a vector (δ, x) in P^2 cannot encode a non feasible schedule. So there is no need to add non-overlapping inequalities, and hence we do not have to provide a separation algorithm or to only consider the extreme points of P^2 .

Finally, a compact formulation for the unrestrictive common due date problem defined by the penalties (α, β) is:

$$(F2) \quad \begin{array}{ll} \min & h_{\alpha,\beta}^{\rho,\sigma}(\delta, x) \\ \text{s.t.} & (\delta, x) \in \text{int}_\delta(P^2) \end{array}$$

where ρ and σ are pre-computed.

Note that polyhedron P^2 does not depend on ρ or σ . Indeed, it is an extended polytope of the classical cut polytope for the complete undirected graph on J [3]. A linear transformation of this polytope has been studied in [17]. From this work we can directly derive that P^2 is a full-dimensional polytope and that inequalities (13-16) define facets of P^2 .

5 General case

In this section, we provide a formulation for the general case based on the ideas of the formulation (F1). In the general case, we have to consider arbitrary earliness unit penalties, that is α_j may be zero for some tasks j in J . We can no longer derive an optimal solution from the one obtained for the instance which does not include these tasks. Indeed, the due date could not allow to add these tasks at the beginning of the schedule. For some instances, such tasks are tardy in all optimal schedules. For example if $J = \llbracket 1, 3 \rrbracket$, $d = 6$, $p_1 = 5$, $p_2 = 3$, $p_3 = 2$, $\alpha_1 = 0$, $\beta_1 = 1$ and $\alpha_2 = \beta_2 = \alpha_3 = \beta_3 = 2$, then the optimal schedule is given by $C_2 = 4 \leq d$, $C_3 = 6 = d$, $C_1 = 11 \geq d$. Note that, conversely, tasks with a zero unit tardiness penalty can still be added at the end of an optimal schedule obtained for the instance reduced to the non-zero earliness penalty tasks in order to obtain an optimal schedule for the original instance. Hence, for the general case, we will set $\alpha \in (\mathbb{R}_+)^J$ and $\beta \in (\mathbb{R}_+^*)^J$.

5.1 Dominance properties

In the general case, the dominance of the d -blocks is no longer valid. Let us define a **d -or-left-block** as a block which is a d -schedule or which starts at time 0, or both, to enunciate the following dominance property in this case [11].

Lemma 17

In the general case, d -or-left-blocks are dominant when minimizing $f_{\alpha, \beta}$.

In the sequel, only d -or-left-blocks will be considered.

Due to the potential occurrence of a straddling task in all optimal schedules for some instances, the partition between early and tardy tasks is no longer sufficient to deduce an optimal schedule. As explained in Section 4, we can compute the best d -block with respect to this partition. Conversely computing the best left-block (*i.e.* the best block starting at time 0) with respect to this partition is not straightforward, since we cannot say *a priori* which is the straddling task among the tardy ones.

Let us consider the best left-block with respect to a given partition. Then the time a between the beginning of the straddling task and d is equal to $d - p(E)$ and the straddling task belongs to $\{j \in T \mid p_j > a\}$, where E (resp. T) denotes the set of early (resp. tardy) tasks given by the partition. One can conjecture that the straddling task maximizes β_j/p_j over this set. However, it is not the case, as we shown by the following instance:

$$\begin{aligned} J = \llbracket 1, 8 \rrbracket, \quad d = 2, \quad \forall i \in [1..6], \quad p_i = 1, \quad \alpha_i = 20, \quad \beta_i = 4, \quad \frac{\beta_i}{p_i} = 4, \\ p_7 = 3, \quad \alpha_7 = 20, \quad \beta_7 = 5, \quad \frac{\beta_7}{p_7} = \frac{5}{3}, \\ p_8 = 4, \quad \alpha_8 = 20, \quad \beta_8 = 8, \quad \frac{\beta_8}{p_8} = 2 \end{aligned}$$

We can easily verify that the optimal partition is $E = \emptyset$, $T = J$, then $a = 2$. According to Lemma 16, an optimal schedule can be found among the left-blocks starting by task 7 and ending by task 8, or starting by task 8 and ending by task 7. The order of the other tasks is arbitrary, since they all have the same ratio. Figure 4 represents one optimal schedule of each type.

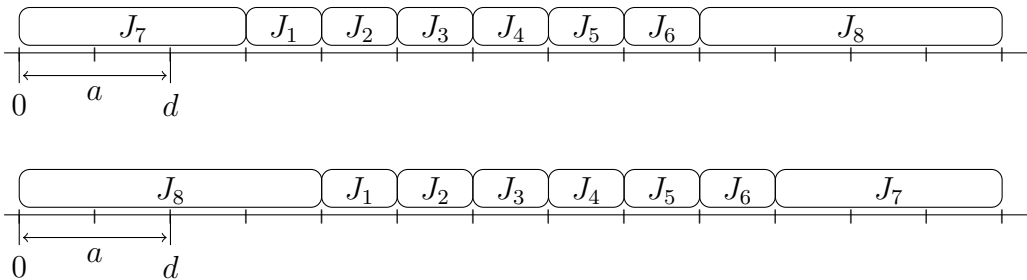


Figure 4: The two types of dominant schedules for $E = \emptyset$ and $T = J$.

The best ones are those starting by task 7 and ending by task 8. Nevertheless the ratio $\beta_7/p_7 = 5/3$ is smaller than the ratio $\beta_8/p_8 = 2$.

This example can be extended to an example where $E \neq \emptyset$ by adding tasks with zero unit earliness penalty and large unit tardiness penalty.

In this example, the non optimality seems to be induced by an incorrect ratio choice: if we consider the ratio $\beta_j/(p_j - a)$ instead of β_j/p_j , task 7 has a greater ratio than task 8. Then one can conjecture that the straddling task j maximizes $\beta_j/(p_j - a)$ over tardy tasks with a processing time larger than a .

Unfortunately, this is also false, as shown by the following instance:

$$\begin{aligned} J = [1, 5], \quad d = 2, \quad \forall i \in [1..3], \quad p_i = 1, \quad \alpha_i = 10, \quad \beta_i = 2, \\ p_4 = 4, \quad \alpha_4 = 10, \quad \beta_4 = 5, \\ p_5 = 3, \quad \alpha_5 = 10, \quad \beta_5 = 3, \end{aligned}$$

We can easily verify that the optimal partition is $E = \emptyset$, $T = J$, then $a = 2$. According to Lemma 16, an optimal schedule can be found among the left-blocks starting by task 4 and ending by task 5, or starting by task 5 and ending by task 4. The order of the other tasks is arbitrary, since they all have the same ratio. Figure 5 represents one optimal schedule of each type.

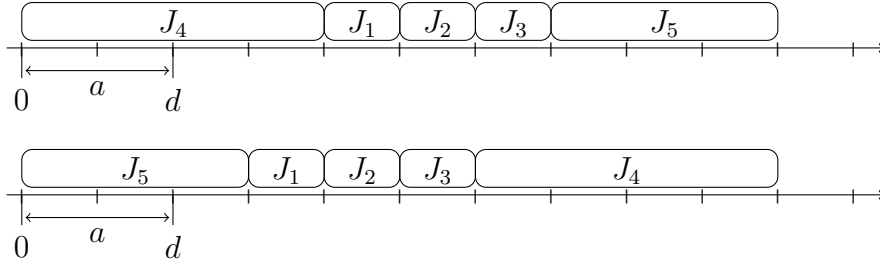


Figure 5: The two types of dominant schedules for $E = \emptyset$ and $T = J$.

The best ones are those starting by task 4 and ending by task 5. Nevertheless the ratio $\beta_4/(p_4 - a) = 2.5$ is smaller than the ratio $\beta_5/(p_5 - a) = 3$. This example can also be extended to an example where $E \neq \emptyset$.

The idea of the compact formulation (F2) for the unrestrictive case was to obtain the value $b(E, T)$ of a best schedule for a fixed partition between early and tardy tasks (E, T) . In the general case, to derive $b(E, T)$ from a partition (E, T) which is feasible (*i.e.* such that $p(E) \leq d$), we have to consider several cases before using the dominance property.

Firstly, if we assume that $b(E, T)$ is achieved by a schedule having an on-time task, then we simply obtain $b(E, T)$ as for the unrestrictive case. Secondly, if we assume that $b(E, T)$ is achieved by a schedule having a straddling task, then we can also assume, without loss of generality, that the schedule starts at time 0 (using Lemma 17). We have then to consider the case where the straddling task is j for each $j \in T$ such that $p_j \geq d - p(E)$. In each case, Lemma 16 allows to derive the optimal schedule and we obtain the value $b(E, T)$ in a similar way as for the unrestrictive case.

But, it seems difficult to derive a linear function from this observation. Therefore, we adapt the first formulation and not the second for the general case.

5.2 A natural formulation for the general case

- *An encoding based on a new reference point*

In case of a schedule with a straddling task j_s , *i.e.* $C_{j_s} - p_{j_s} < d < C_{j_s}$, the tardiness of tardy tasks do not satisfy the non-overlapping constraints, *i.e.* t/T does not satisfy inequalities (Q0), particularly $t_{j_s} > p_{j_s}$. Indeed, these tardiness no longer play the same role as completion times. Therefore, we will use variables describing the schedule with respect to a new reference point, which is the starting time of j_s instead of the due date d .

We introduce a new variable a , so that $d-a$ is the starting time of j_s . The schedule is then a $(d-a)$ -schedule. For each task j in J , we consider a variable e'_j (resp. t'_j) instead of e_j (resp. t_j), representing the earliness (resp. the tardiness) according to the new reference point $d-a$. Figure 6 illustrates this encoding for a schedule holding a straddling task.

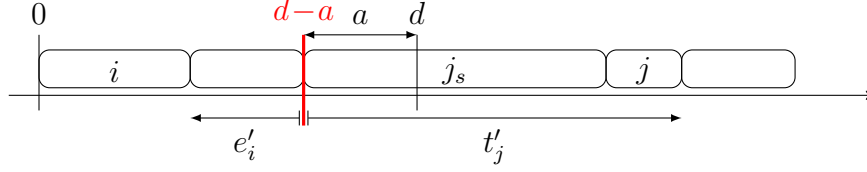


Figure 6: The (a, e', t') encoding for a schedule holding a straddling task j_s

Since we do not know *a priori* if there is a straddling task in the optimal schedule, our formulation must also handle d -blocks. Hence, we also need to encode d -blocks by variables a, e', t' .

In case of a schedule holding an on-time task j_t , we can keep d as the reference point, since we can use earliness and tardiness as proposed in formulation (F1). Hence, the first encoding consists in setting $a=0$, and using e' (resp. t') to represent earliness (resp. tardiness). Figure 7 illustrates this encoding for a schedule holding an on-time task.

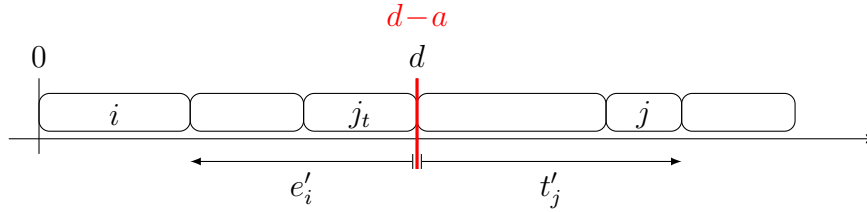


Figure 7: The first (a, e', t') encoding for a schedule holding an on-time task j_t

Unfortunately, to ensure that a takes the expected value in case of a schedule holding a straddling task, we will introduce a boolean variable to identify the task j_0 beginning at $d-a$. It force to have in every schedule a task beginning at $d-a$. Therefore, this first encoding is not valid in case of a d -block without tardy task. We then propose a second encoding for the d -blocks. It consists in choosing the starting time of j_t as the new reference point, which is setting $a=p_{j_t}$. This second encoding can be used also for a schedule holding an on-time task and having tardy tasks, as illustrated by Figure 8.

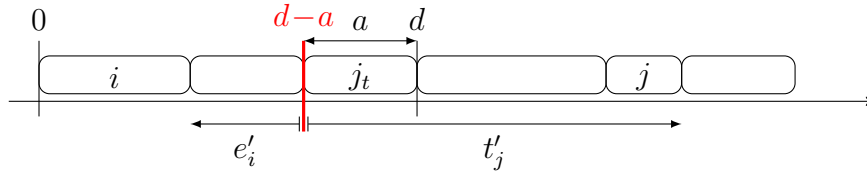


Figure 8: The second (a, e', t') encoding for a schedule holding an on-time task j_t

To sum up, the first encoding, with $a=0$, is suitable for d -blocks, except those without tardy tasks. Conversely, the second encoding, with $a=p_{j_t}$, is suitable for any d -block. Fortunately, the three encodings proposed in this section can be decoded in the same way : $C = (d-a-e'_j+t'_j)_{j \in J}$ gives the completion times of the encoded schedule.

- *Consistency between e' and t' using δ variables*

To ensure consistency between e' and t' , we use again variables δ . Whereas in the previous formulation δ_j indicated if task j completes before or at d , in this formulation δ_j indicates if the task completes before or

at $d-a$. We also use inequalities (5-8) where e (resp. t) are replaced by e' (resp. t'). These inequalities will be denoted (5'-8') in the sequel.

Note that δ_j indicates no longer necessarily if task j is early or not. Keeping the previous notations $E(\delta) = \{j \in J \mid \delta_j = 1\}$ and $T(\delta) = \{j \in J \mid \delta_j = 0\}$, $(E(\delta), T(\delta))$ is not the partition between early and tardy tasks as soon as we use the second encoding for a d -block. Therefore, we introduce a new partition of tasks: if C encodes a schedule by its completion times, we define

$$\tilde{E}(C) = \{j \in J \mid C_j < d\} \quad \text{and} \quad \tilde{T}(C) = \{j \in J \mid C_j \leq d\}$$

Note that if there is a straddling task in the schedule, then $E(C) = \tilde{E}(C)$ and $T(C) = \tilde{T}(C)$. Besides in this case, the only encoding is such that $E(\delta) = E(C) = \tilde{E}(C)$ and $T(\delta) = T(C) = \tilde{T}(C)$.

In the case of a d -block, using the first encoding we also have $E(\delta) = E(C)$ and $T(\delta) = T(C)$, but using the second one we have $E(\delta) = \tilde{E}(C)$ and $T(\delta) = \tilde{T}(C)$.

- *Handling the positivity*

Since the due date can be smaller than $p(J)$, avoiding overlapping and idle times does not ensure the positivity constraint. Therefore, we add the following inequalities ensuring that $e'_j + p_j \leq d-a$ for each task j completing before $d-a$. They are valid since d is an upper bound of a .

$$\forall j \in J, \quad e'_j + p_j \delta_j \leq d-a \tag{17}$$

- *Handling the non-overlapping*

To ensure the non overlapping, we use again variables x , satisfying (13-16) and the inequalities (Q1) and (Q2), where e (resp. t) are replaced by e' (resp. t'). These inequalities will be denoted (Q1') and (Q2') in the sequel.

In order to ensure that tasks completing before or at $d-a$ do not overlap using inequalities (Q1'), inequalities (17) must not restrict too much e'_j from above.

Indeed, an inequality of the form $C_j \leq M$ is compatible with the non-overlapping inequalities (Q0) only if $M \geq p(J)$. If $M < p(J)$, adding such an inequality makes appear extreme points which can be reached by minimization, whereas they do not correspond to feasible schedules.

For example if $J = \llbracket 1, 2 \rrbracket$, $d = 5$, $p_1 = p_2 = 3$, $\alpha_1 = \alpha_2 = 1$, $\beta_1 = \beta_2 = 10$, then the schedule where both tasks are completed at time 3, which is not feasible, corresponds to an extreme point where $a = 0$, $e'_1 = e'_2 = d-a$. Moreover, this schedule induces a total penalty of 4, while the penalty of an optimal schedule is 12. The extremality of this unfeasible schedule comes from the fact that $d-a$ restricts too much the value of e' . Indeed, we have $d-a = 5$ and $p(E) = 6$, while we should have $d-a \geq p(E)$.

Therefore, we introduce the following inequalities:

$$\sum_{j \in J} p_j \delta_j \leq d-a \tag{18}$$

In order that inequalities (Q1') (resp. (Q2')) guarantee that the tasks completing before (resp. after) $d-a$ do not overlap, the total penalty must be a decreasing function of variable e'_j (resp. t'_j) for each task j such that $\delta_j = 1$ (resp. $\delta_j = 0$). If a is such that $d-a$ is the starting time of the straddling task, the on-time task or the first tardy task as proposed by the previous encodings, these two constraints are ensured. But we have to ensure that the variable a takes the expected value by linear inequalities.

- *Ensuring that a takes the expected value*

In spite of their apparent symmetry, the two previous constraints on $d-a$ are completely different.

To ensure the first one, it suffices to ensure that any task completing before or at $d-a$ completes before or at d . Indeed, reducing e'_j for such a task j while satisfying the inequality (Q1') associated to $\{j\}$, i.e. $e'_j \geq 0$, task j remains early and its tardiness decreases, which reduces the induced penalty. Therefore, the first constraint is guaranteed by the following inequality:

$$a \geq 0 \tag{19}$$

To ensure the second one, ensuring that any task completing after $d-a$ completes after or at d is not sufficient. Indeed, reducing t'_j for such a task j while satisfying the inequality (Q2') associated to $\{j\}$, *i.e.* $t'_j \geq p_j$, task j can become early, so the induced penalty does not necessarily decrease. Figure 9 illustrates the extreme case of this phenomenon, that is when $a=d$, $E(\delta)=\emptyset$, and all early tasks overlap each others to be on-time.

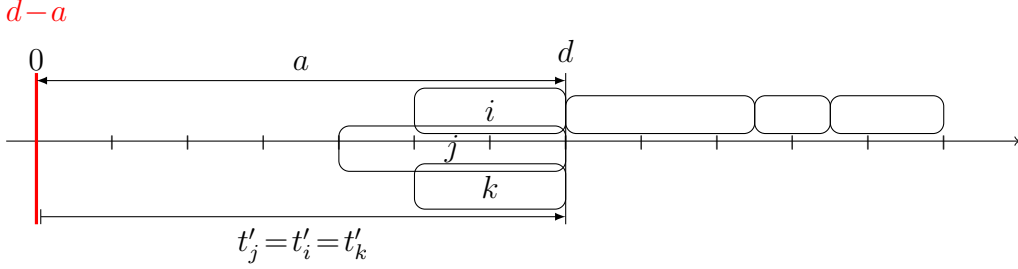


Figure 9: An unfeasible schedule when $a=d$

Note that this case appears even if we add inequalities $\forall j \in J, t'_j \geq a(1-\delta_j)$. Adding inequalities $\forall j \in J, p_j \geq a(1-\delta_j)$, could avoid this issue, but unfortunately they are not valid, since a task completing after $d-a$ can be shorter than a , as longer it is not the first one.

In order to ensure that the first task j_0 completing after $d-a$ completes after or at d , we introduce a boolean variable γ_j for each task j , representing if j is j_0 , and the following inequalities:

$$\sum_{j \in J} \gamma_j = 1 \quad (20)$$

$$\forall j \in J, \quad \delta_j \leq 1 - \gamma_j \quad (21)$$

$$\forall j \in J, \quad t'_j \leq p_j + (1 - \gamma_j)(p(J) - p_j) \quad (22)$$

Whereas inequalities (20-21) ensure that γ designates one and only one task i_0 among those completing after $d-a$, inequalities (22) ensure that i_0 is the first one, *i.e.* $i_0 = j_0$. Indeed, they ensure that $t'_{i_0} \leq p_{i_0}$, and since $t'_{i_0} \geq p_{i_0}$ by inequality (Q2') associated to the singleton $\{i_0\}$, we deduce that $t'_{i_0} = p_{i_0}$. Then, the inequality (Q2') associated to a pair $\{i_0, j\}$ with $\delta_j = 0$, suffices to prove that task j completes after i_0 .

Lemma 18

Let $(t', \delta, x, \gamma) \in \mathbb{R}^J \times \{0, 1\}^J \times [0, 1]^{J^<} \times [0, 1]^J$.
(i) $\gamma \in \{0, 1\}^J$ and γ and δ satisfy (20-21) $\Leftrightarrow \exists i_0 \in T(\delta), \gamma = \mathbb{1}_{i_0}$
(ii) In case (i) holds and t', δ, x satisfy (13-16), (22) and (Q2'),
then $t'_{i_0} = p_{i_0}$ and $\forall j \in T(\delta), j \neq i_0, t'_j \geq t'_{i_0} + p_j$.

Using γ , which identify j_0 , we add the following valid inequalities to ensure that $a \leq p_{j_0} = t'_{j_0}$.

$$\forall j \in J, \quad a \leq p_j + (1 - \gamma_j)d \quad (23)$$

- A linear objective function using e', t', a and b variables

Using e' and t' variables instead of e and t offers an easy way to ensure positivity, consistency and non-overlapping at the expense of a linearization of the product $a\delta_j$. Indeed, in the objective function, we need a linear expression for the earliness (resp. the tardiness) of any task j in J , which is equal to $e'_j + a\delta_j$ (resp. to $t'_j - a(1-\delta_j)$).

Then we introduce a variable b_j for each task j in J to replace the product $a\delta_j$. We add the following inequalities to ensure that b variables take the expected values.

$$\forall j \in J, \quad b_j \geq 0 \quad (24)$$

$$\forall j \in J, \quad b_j \leq a \quad (25)$$

$$\forall j \in J, \quad b_j \leq \delta_j d \quad (26)$$

$$\forall j \in J, \quad t'_j \geq a - (1 - \delta_j)d \quad (27)$$

Since d is an upper bound of a by construction, we get the following lemma:

Lemma 19

Let $(a, b, \delta) \in \mathbb{R} \times \mathbb{R}^J \times \{0, 1\}^J$.
 a, b and δ satisfy inequalities (24-27) $\Leftrightarrow b = a\delta$

Then the total penalty of a schedule encoded by (e', t', a, b) is :

$$h_{\alpha, \beta}(e', t', a, b) = \sum_{j \in J} \alpha_j e'_j + \beta_j t'_j + (\alpha_j + \beta_j) b_j - \beta_j a$$

which is linear. If C encodes a schedule by its completion times, the two possible vectors (e', t', a, b) encoding this schedule are:

$$\theta'(C) = \left(([d - a - C_j]^+)_{j \in J}, ([C_j - (d - a)]^+)_{j \in J}, a, a \mathbb{1}_{E(C)} \right) \text{ where } a = d - \min_{i \in T(C)} C_i - p_i$$

$$\tilde{\theta}'(C) = \left(([d - \tilde{a} - C_j]^+)_{j \in J}, ([C_j - (d - \tilde{a})]^+)_{j \in J}, \tilde{a}, \tilde{a} \mathbb{1}_{\tilde{E}(C)} \right) \text{ where } \tilde{a} = d - \min_{i \in \tilde{T}(C)} C_i - p_i$$

Note that if the schedule holds a straddling task, then $\theta(C) = \tilde{\theta}'(C)$, since $E(C) = \tilde{E}(C)$ and $T(C) = \tilde{T}(C)$. Even for a schedule admitting two different encodings, (*i.e.* for a d -schedule with at least one tardy task) the function $h_{\alpha, \beta}$ holds the total penalty, as long as the schedule satisfies the non-overlapping constraint:

Lemma 20

Let $C \in \mathbb{R}^J$. **If** C satisfies (1), **then** $h_{\alpha, \beta}(\theta'(C)) = h_{\alpha, \beta}(\tilde{\theta}'(C)) = f_{\alpha, \beta}(\theta'(C))$.

- *Formulation (F3)*

Let us define the polyhedron:

$$P^3 = \left\{ (e', t', \delta, x, a, b, \gamma) \in \mathbb{R}^J \times \mathbb{R}^J \times [0, 1]^J \times \mathbb{R}^{J^<} \times \mathbb{R} \times \mathbb{R}^J \times [0, 1]^J \mid \begin{array}{l} (5'-8'), (13-16), (17-18), (23-19), (24-27), \\ (20-22), (Q1'), \text{ and } (Q2') \text{ are satisfied} \end{array} \right\}$$

Note that this polyhedron depends on d , in addition to p .

Inequalities (Q1') and (Q2') require the same separation algorithm as for (Q1) and (Q2), which will be developed in Section 6.

In order to keep only points with integer δ and γ , we introduce the operator $\mathbf{int}_{\delta, \gamma}$:

$$\mathbf{int}_{\delta, \gamma}(V) = \{ (e', t', \delta, x, a, b, \gamma) \in V \mid \delta \in \{0, 1\}^J, \gamma \in \{0, 1\}^J \}$$

for V included in $\mathbb{R}^J \times \mathbb{R}^J \times \mathbb{R}^J \times \mathbb{R}^{J^<} \times \mathbb{R} \times \mathbb{R}^J \times \mathbb{R}^J$.

Finally, our formulation for the general common due date problem is :

$$(F3) \quad \begin{array}{ll} \min & h_{\alpha, \beta}(e', t', a, b) \\ \text{s.t.} & (e', t', \delta, x, a, b, \gamma) \in \mathbf{int}_{\delta, \gamma}(\mathbf{extr}(P^3)) \end{array}$$

5.3 Validity of Formulation (F3)

Thanks to the natural variables e' and t' , ensuring the non-overlapping constraint reduces to ensuring the positivity and non-overlapping constraints for two subsets of tasks. In contrast with Formulation (F1) where these two subsets are the early and the tardy tasks (*cf.* Lemma 8), in Formulation (F3), the subsets to consider depend on the occurrence of a straddling or an on-time task, as detailed in the following lemma.

Lemma 21

Let $C \in \mathbb{R}^J$.

(i) **If** there exists $j_s \in J$ such that $C_{j_s} - p_{j_s} < d < C_{j_s}$ and $(e', t', a, b) = \theta'(C) = \tilde{\theta}'(C)$,

then C satisfies (1) $\Leftrightarrow \begin{cases} (e' + p)_{/E(C)} & \text{satisfies (0) and (1)} \\ t'_{/T(C)} & \text{satisfies (0) and (1)} \end{cases}$

(ii) **If** there exists $j_t \in J$ such that $C_{j_t} = d$ and $(e', t', a, b) = \tilde{\theta}'(C)$,

then C satisfies (1) $\Leftrightarrow \begin{cases} (e' + p)_{/\tilde{E}(C)} & \text{satisfies (0) and (1)} \\ t'_{/\tilde{T}(C)} & \text{satisfies (0) and (1)} \end{cases}$

The following theorem establishes that a feasible schedule, under some assumptions, is encoded by an integer point of P^3 . In particular a d -or-left-block is encoded by an integer point of P^3 .

Theorem 22

Let $C \in \mathbb{R}^J$ satisfying (0) and (1).

(i) **If** there exists $j_s \in J$ such that $C_{j_s} - p_{j_s} < d < C_{j_s}$ and $\forall j \in J, d - p(J) \leq C_j - p_j$ and $C_j \leq C_{j_s} - p_{j_s} + p(J)$, **then** there exists $X = (e', t', \delta, x, a, b, \gamma) \in \text{int}_{\delta, \gamma}(P^3)$, such that $\theta'(C) = (e', t', a, b)$.

(ii) **If** there exists $j_t \in J$ such that $C_{j_t} = d$ and $\forall j \in J, d - p(J) \leq C_j - p_j$ and $C_j \leq C_{j_t} - p_{j_t} + p(J)$, **then** there exists $X = (e', t', \delta, x, a, b, \gamma) \in \text{int}_{\delta, \gamma}(P^3)$, such that $\tilde{\theta}'(C) = (e', t', a, b)$.

Proof: Let us start by proving (i).

From C , let us set: $(e', t', a, b) = \theta'(C)$, $\delta = \mathbb{1}_{E(C)}$, $x = (\mathbb{1}_{\delta_i \neq \delta_j})_{(i,j) \in J \times J}$, $\gamma = \mathbb{1}_{j_s}$ and $X = (e', t', \delta, x, a, b, \gamma)$.

We will prove that $X \in \text{int}_{\delta, \gamma}(P^3)$.

Note that the definition of δ ensures that $\delta \in \{0, 1\}^J$, and that $E(\delta) = E(C)$ and $T(\delta) = T(C)$, which allows the notation E and T for sake of brevity. By Lemma 9(i), the definition of x ensures that inequalities (13-16) are satisfied. By Lemma 18(i), the definition of γ ensures that inequalities (20-21) are satisfied, since $j_s \in T$. By Lemma 19, inequalities (24-27) are satisfied, since $b = a \mathbb{1}_{E(C)} = a \delta$.

For the straddling task j_s , we have $C_{j_s} - p_{j_s} = \min_{j \in T(C)} C_j - p_j$ then $a = d - (C_{j_s} - p_{j_s})$, by definition of θ' .

Since task j_s starts at or after 0 and before d , i.e. $0 \leq C_{j_s} - p_{j_s} < d$, we have $0 < a \leq d$. Thus inequality (19) is satisfied, and for any task $j \neq j_s$, $a \leq d + p_j = (1 - \gamma_j) d + p_j$. More precisely, task j_s starts after all early tasks, and since they do not overlap, $p(E) \leq C_{j_0} - p_{j_0} = d - a$, thus inequality (18) holds. Since task j_s completes after d , i.e. $C_{j_s} > d$, we get $a = p_{j_s} + (d - C_{j_s}) < p_{j_s} = p_{j_s} + (1 - \gamma_{j_s}) d$. We deduce that inequalities (23) are satisfied.

Inequalities (5') and (7'), are satisfied by construction of e' and t' .

For a task j in E , $C_j \leq C_{j_0} - p_{j_0} = d - a$ since j and j_0 do not overlap, then $e'_j = d - a - C_j$ and $t'_j = 0$. The corresponding inequality (8') is thus satisfied, as well as (22) since $p_j + (1 - \gamma_j)(p(J) - p_j) = p(J) \geq 0$. By assumption $C_j \geq d - p(J)$, thus $e'_j \leq d - a - (d - p(J)) \leq p(J)$, and inequality (6') is also satisfied for j . Moreover, $d - e'_j - p_j \delta_j = a + C_j - p_j$, and by positivity constraint $C_j - p_j \geq 0$, thus $d - e'_j - p_j \delta_j \geq a$ and inequality (17) is satisfied for j .

For a task j in T , $C_j \geq d \geq d - a$, then $e'_j = 0$ and $t'_j = C_j - d - a$. The corresponding inequality (6') is thus satisfied. Moreover, $d - e'_j - p_j \delta_j = d \geq a$, then inequality (17) is satisfied for j . By assumption $C_j \leq C_{j_0} - p_{j_0} + p(J) \leq d + p(J)$, thus $t'_j \leq (d + p(J)) - d - a$ and then $t'_j \leq p(J)$. We deduce that the corresponding inequality (8') is also satisfied, as well as inequality (22), since $p_j + (1 - \gamma_j)(p(J) - p_j)$ is equal to $p(J)$ (resp. to $p_{j_s} = C_{j_0} - d - a = t'_{j_s}$) if $j \neq j_s$ (resp. if $j = j_s$).

Since C encodes a feasible schedule, C satisfies (1). Using Lemma 21, $(e' + p)_{/E}$, as well as $t'_{/T}$, satisfies (0) and (1). Applying Property 1 to these two vectors, we deduce that they satisfy (Q0), and using Lemma 10, that e', δ, x satisfy (Q1') and t', δ, x satisfy (Q2'). Thus, X belongs to $\text{int}_{\delta, \gamma}(P^3)$.

Rewriting the proof by replacing θ' by $\tilde{\theta}'$, $E(C)$ by $\tilde{E}(C)$, $T(C)$ by $\tilde{T}(C)$, and the straddling task j_s by the on-time task j_t provides almost the proof of (ii). The only difference lies in the justification of

inequality (23) for j_t : in this case $C_{j_t} = d$, then $a = p_{j_s} + (d - C_{j_s}) = p_{j_s} = p_{j_s} + (1 - \gamma_{j_s})d$. \square

The following theorem establishes that an optimal solution of formulation (F3) is a solution for the general common due date problem.

Theorem 23

Let $X^* = (e', t', \delta, x, a, b, \gamma) \in \text{int}_{\delta, \gamma}(P^3)$.

If $\alpha \in (\mathbb{R}_+^*)^J$, $X^* \in \text{extr}(P^3)$ and (e', t', a, b) minimizes $h_{\alpha, \beta}$ **then** X^* encodes a d -or-left-block, by θ' or $\tilde{\theta}'$.

Proof: Let us set, for any task j in J , $C_j^* = (d - a) - e'_j + t'_j$.

The first step of the proof is to show that C^* gives the completion times of the schedule encoded by X^* using θ' or $\tilde{\theta}'$ i.e. that $X^* = \theta'(C^*)$ or $X^* = \tilde{\theta}'(C^*)$.

First we derive from inequalities (5'-8') that $\forall j \in T(\delta)$, $e'_j = 0$ and $\forall j \in E(\delta)$, $t'_j = 0$.

Since δ and γ are in $\{0, 1\}^J$, and X^* satisfies (13-16), (20-21) and (Q2'), Lemma 18 ensures that there exists $j_0 \in T(\delta)$ such that $\gamma = \mathbb{1}_{\{j_0\}}$, $t'_{j_0} = p_{j_0}$, and $\forall j \in T(\delta)$, $j \neq j_0$, $t'_j \geq t'_{j_0} + p_j$. Since j_0 is in $T(\delta)$, $e'_{j_0} = 0$ and $C_{j_0}^* - p_{j_0} = d - a$. Then for any other task j in $T(\delta)$, $C_j^* - p_j = (C_{j_0}^* - p_{j_0}) + t'_j - p_j \geq (C_{j_0}^* - p_{j_0}) + t'_{j_0} = C_{j_0}^* > C_{j_0}^* - p_{j_0}$. We deduce that $C_{j_0}^* - p_{j_0} = \min_{j \in T(\delta)} C_j^* - p_j$, and then $a = d - \min_{j \in T(\delta)} C_j^* - p_j$.

The question is whether $T(\delta) = T(C^*)$ or $T(\delta) = \tilde{T}(C^*)$. Indeed, if $T(\delta) = T(C^*)$, the value of a is the one expected with the encoding θ' , whereas if $T(\delta) = \tilde{T}(C^*)$, it is the one expected with $\tilde{\theta}'$.

For any task $j \neq j_0$ in $T(\delta)$, $C_j^* = d - a + t'_j > d - a + t'_{j_0} = d - a + p_{j_0}$. Since $\gamma_{j_0} = 1$, inequality (23) gives $a \leq p_{j_0}$, thus $C_j^* > d$. We deduce that $T(\delta) \setminus \{j_0\} \subseteq T(C^*)$. Conversely, for a task j in $T(C^*)$, $C_j^* > d$, which is equivalent to $t'_j - e'_j > a$. Since $a \geq 0$ by inequality (19), $t'_j > e'_j$, which would be impossible if j was in $E(\delta)$, according to inequalities (5') and (8'). We deduce that $T(C^*) \subseteq T(\delta)$. Two cases have to be considered:

→ If $a < p_{j_0}$, then $C_{j_0}^* > d$, i.e. $j_0 \in T(C^*)$, and then $T(\delta) = T(C^*)$ and $E(\delta) = E(C^*)$.

→ If conversely $a = p_{j_0}$, then $C_{j_0}^* = d$ and $j_0 \in \tilde{T}(C^*)$, we deduce that $T(\delta) \subseteq \tilde{T}(C^*)$. For j in $\tilde{T}(C^*)$, either $j \in T(C^*) \subseteq T(\delta)$ or $C_j = d$, that is $t'_j = e'_j + a = e'_j + p_{j_0} > e'_j$, and necessarily $j \in T(\delta)$. We conclude that $T(\delta) = \tilde{T}(C^*)$ and $E(\delta) = \tilde{E}(C^*)$.

For the remainder of the proof, we assume that we are in the first case. Then E (resp. T) will denote $E(C^*)$ (resp. $T(C^*)$), and we will use the encoding θ' . To handle the second case, it suffices to replace $E(C^*)$ by $\tilde{E}(C^*)$, $T(C^*)$ by $\tilde{T}(C^*)$, and θ' by $\tilde{\theta}'$ in the second step, and using that j_0 is an on-time task in the third step.

We can rewrite δ as $\mathbb{1}_{E(C^*)}$, and thus b as $a\mathbb{1}_{E(C^*)}$, since $b = a\delta$ by inequalities (24-27) and Lemma 19. Using inequalities (5'-8'), it is easy to show that $e' = ([d - a - C_j^*]^+)$ and $t' = ([C_j^* - (d - a)]^+)$. Then we can conclude that $(e', t', a, b) = \theta'(C^*)$, that is that C^* and (e', t', a, b) encode the same schedule, which will be denoted by \mathcal{S}^* .

The second step is to show that \mathcal{S}^* is feasible, by proving that C^* satisfies (0) and (1).

For a task j in E , inequality (17) ensures that $p_j \leq d - a - e'_j = C_j^*$. For a task j in T , inequality (17) ensures that $a \leq d$, then $C_j = d - a + t'_j \geq t'_j$. For another, we deduce that $t'_j \geq p_j$ from inequality (Q2') associated to $\{j\}$. Thus C^* satisfies (0).

To show that C^* satisfies (1), using Lemma 21, it remains to show that vectors $(e' + p)_{/E}$ and $t'_{/T}$ satisfy (0) and (1). Since inequalities (Q1) and (Q2) are satisfied, we know from Lemma 10 that $(e' + p)_{/E}$ and $t'_{/T}$ satisfy inequalities (Q0). On one hand, these inequalities for the singletons ensure that both vectors satisfy (0). On the other hand, inequalities (Q0) will allow us to show that both vectors satisfy (1).

Let us assume that $(e' + p)_{/E}$ does not satisfy (1). Then there exist two tasks i and j in E such that $e'_i + p_i \leq e'_j + p_j < (e'_i + p_i) + p_j$. Three cases have to be considered:

→ If $e'_j + p_j \geq p(J)$, then $e'_j + p_j \geq p(E)$. Applying Lemma 4, we can construct a vector e'^- such that $X^- = (e'^-, t', \delta, x, a, b, \gamma)$ is in $\text{int}_{\delta, \gamma}(P^3)$ and $h_{\alpha, \beta}(e'^-, t', a, b) < h_{\alpha, \beta}(e', t', a, b)$ since $\alpha \in ((\mathbb{R}_+^*)^J)$, which contradicts the minimality of (e', t', a, b) .

→ If $e'_j + p_j = d - a$, we can derive the same contradiction since $d - a \geq p(E)$ from inequality (18).

→ If $e'_j + p_j < p(J)$ and $e'_j + p_j < d - a$, then applying Lemma 3 to $(e' + p)/E$, we can construct two vectors e'^{+-} and e'^{-+} such that $X^{+-} = (e'^{+-}, t', \delta, x, a, b, \gamma)$ and $X^{-+} = (e'^{-+}, t', \delta, x, a, b, \gamma)$ are in $\text{int}_{\delta, \gamma}(P^3)$ and that X^* is the middle point of the segment $[X^{+-}, X^{-+}]$. which contradicts the extremality of X^* .

Similarly, let us assume that $t'_{/T}$ does not satisfy (1). Then there exist two tasks i and j in T such that $t'_i \leq t'_j < t'_i + p_j$. Since $\forall k \in T(\delta)$, $k \neq j_0$, $t'_k \geq t'_{j_0} + p_k$, we deduce that $i \neq j_0$. Then for tasks i and j , inequalities (8') and (22) are equivalent, and t'_i and t'_j are only bounded from above by $p(J)$. Then two cases have to be considered:

→ If $t'_j \geq p(J)$, then $t'_j \geq p(T)$. Applying Lemma 4, we can derive a contradiction to the minimality of (e', t', a, b) .

→ If $t'_j < p(J)$, Applying Lemma 3 we can derive a contradiction to the extremality of X^* .

Finally, \mathcal{S}^* is feasible.

The third step is to show that \mathcal{S}^* is a d -or-left-block.

We first prove that \mathcal{S}^* is a block using the same method as in the proof of Theorem 13. Assuming that \mathcal{S}^* is not a block, we construct a better schedule $\hat{\mathcal{S}}$ by tightening tasks around d . Using Theorem 22, there exists $\hat{X} \in \text{int}_{\delta, \gamma}(P^3)$ encoding $\hat{\mathcal{S}}$, and it contradicts the minimality of (e', t', a, b) .

Thus \mathcal{S}^* is a block. Now we have to show that \mathcal{S}^* starts at time 0 or holds an on-time task. Let us assume that it is not the case, then setting $\varepsilon = \frac{1}{2} \min(p_{j_0} - a, a, s)$ where s denotes the starting time of \mathcal{S}^* , we have $\varepsilon > 0$. Setting $a^- = a - \varepsilon$ (resp. $a^+ = a + \varepsilon$), and $X^- = (e', t', \delta, x, a^-, a^- \delta, \gamma)$ (resp. $X^+ = (e', t', \delta, x, a^+, a^+ \delta, \gamma)$), X^- (resp. X^+) encodes using θ' the schedule obtained by shifting backward (resp. forward) by ε time unit all the tasks. By definition of ε , X^- (resp. X^+) still satisfies inequalities (17), (23), (19) and (18), thus $X^- \in P^3$ (resp. $X^+ \in P^3$). Since X^* is the middle of $[X^-, X^+]$, that contradicts the extremality of X^* .

We deduce that X^* encodes a d -or-left-block. □

If some tasks have zero unitary earliness penalty, formulation (F3) provides a vector $X^* = (e', t', \delta, x, a, b, \gamma)$ which partially encodes an optimal schedule. Indeed, except for early tasks having a zero unit earliness penalty, the completion time of a task j is given as previously by $C_j^* = (d - a) - e'_j + t'_j$. Conversely, for an early task j such that $\alpha_j = 0$, e'_j could be $d - p_j$ for instance and the previous encoding would give $C_j^* = p_j$. If there are several early tasks having zero unitary earliness penalty, an overlapping would appear at time 0. Since their unitary earliness penalty is zero, the minimality of X^* does not ensure that these tasks are well spread out (in this context Lemma 4 cannot be applied). However, the minimality of X^* ensures that the other early tasks (*i.e.* having a non-zero unitary earliness penalty) are right-tight with respect to d . Hence, using inequality (18), there is enough time between 0 and their processing duration to process the overlapping tasks. Thus, it suffices to schedule these tasks in an arbitrary order from time 0 to obtain a feasible schedule \mathcal{S} . Since these tasks do not induce any penalty, the total penalty of \mathcal{S} is $h_{\alpha, \beta}(X^*)$, regardless of their order. We deduce that \mathcal{S} is an optimal schedule.

The following theorem establishes that the general common due date problem reduces to solving formulation (F3). We omit the proof since it follows the same lines as the one of Theorem 14.

Theorem 24

Any optimal d -or-left-block, is encoded by a vector minimizing $h_{\alpha, \beta}$ on $\text{int}_{\delta, \gamma}(\text{extr}(P^3))$.
Conversely, any vector minimizing $h_{\alpha, \beta}$ on $\text{int}_{\delta, \gamma}(\text{extr}(P^3))$, encodes an optimal d -or-left-block.

6 Separation algorithms

In this section, we explain how to separate inequalities (Q1),(Q2), (Q1') or (Q2'), by solving a min-cut problem in a suitable graph. We write the following development for inequalities (Q1) and (Q2), but a rewriting exercise suffices to obtain the equivalent results for inequalities (Q1') and (Q2').

Let $X = (e, t, \delta, x) \in \mathbb{R}^J \times \mathbb{R}^J \times [0, 1]^J \times \mathbb{R}^{J^<}$ a vector satisfying inequalities (5-8) and (13-16). The separation problem for inequalities (Q1) is to find a subset S of J such that X does not satisfy the associated inequality (Q1) or to guarantee that X satisfies all inequalities (Q1).

We will first show that this separation problem reduces to the maximization of a set function $\Gamma^{c,q}$ defined from parameters $(c, q) \in \mathbb{R}^J \times \mathbb{R}^{J^<}$ as :

$$\forall S \subseteq J, \Gamma^{c,q}(S) = \sum_{(i,j) \in S^<} q_{i,j} + \sum_{i \in S} c_i$$

Indeed we have:

$$\begin{aligned} X \text{ satisfies (Q1)} &\Leftrightarrow \forall S \subseteq J, \sum_{(i,j) \in S^<} p_i p_j \frac{\delta_i + \delta_j - x_{i,j}}{2} \leq \sum_{i \in S} p_i e_i \\ &\Leftrightarrow \forall S \subseteq J, \sum_{(i,j) \in S^<} p_i p_j (\delta_i + \delta_j - x_{i,j}) - 2 \sum_{i \in S} p_i e_i \leq 0 \\ &\Leftrightarrow \forall S \subseteq J, \Gamma^{c^1, q^1}(S) \leq 0 \end{aligned}$$

where $c^1 = -2(p_j e_j)_{j \in J}$ and $q^1 = (p_i p_j (\delta_i + \delta_j - x_{i,j}))_{(i,j) \in J^<}$. Then it suffices to maximize Γ^{c^1, q^1} over the subsets of J . Indeed, if the obtained value is negative or zero, then X satisfies all inequalities (Q1), conversely if the obtained value is positive, then the maximizing set is not empty and corresponds to an inequality (Q1) that X does not satisfy.

Similarly, the separation problem of inequalities (Q2), is equivalent to the maximization of Γ^{c^2, q^2} where $c^2 = 2((1 - \delta_j)p_j^2 - p_j t_j)_{j \in J}$ and $q^2 = (p_i p_j (2 - (\delta_i + \delta_j) - x_{i,j}))_{(i,j) \in J^<}$.

Note that in both definitions of Γ^{c^1, q^1} and Γ^{c^2, q^2} , the parameter q is positive since δ and x satisfy inequalities (15-16). Therefore, let us now explain how to reduce the maximization of $\Gamma^{c,q}$ for $(c, q) \in \mathbb{R}^J \times (\mathbb{R}_+^*)^{J^<}$ to a min-cut problem in an undirected graph as proposed by Picard and Ratliff [18]. Let us assume that $J = \llbracket 1, n \rrbracket$ for sake of brevity. We consider the weighted undirected graph $G = (V, E, w)$, where $V = \llbracket 0, n+1 \rrbracket$, $E = \{ \{i, j\} \mid (i, j) \in V^2, \{i, j\} \neq \{0, n+1\} \}$ and

$$\forall j \in J, k_j = 2c_j + \sum_{i=1}^{j-1} q_{i,j} + \sum_{k=j+1}^n q_{j,k}, \quad w_{\{0,j\}} = [k_j]^+, \quad w_{\{j,n+1\}} = [k_j]^- \quad \text{and} \quad \forall (i, j) \in J^<, w_{\{i,j\}} = q_{i,j}$$

Figure 10 gives an illustration of such graph for $n=5$.

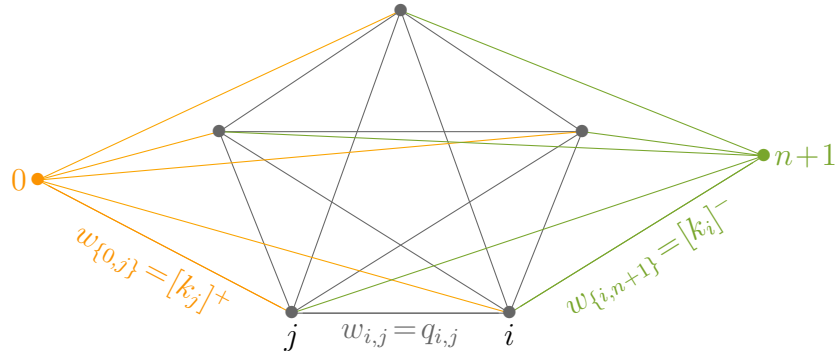


Figure 10: Illustration of the weighted undirected graph G for $n=5$

Note that V and E only depend on J , and w only depends on parameters c and q . For a cut (W, \overline{W}) , *i.e.* $W \cap \overline{W} = \emptyset$ and $W \cup \overline{W} = V$, let $w(W, \overline{W})$ denote its weight according to w , *i.e.* $w(W, \overline{W}) = \sum_{\substack{i \in W \\ j \in \overline{W}}} w_{\{i,j\}}$.

Then for any $S \subseteq J$ we have:

$$\Gamma^{c,q}(S) = -\frac{1}{2} w(S \cup \{0\}, \llbracket 1, n+1 \rrbracket \setminus S) + \frac{Q+C}{2} + \frac{K}{4} \quad \text{where} \quad Q = \sum_{(i,j) \in J^<} q_{i,j}, \quad C = \sum_{j \in J} c_j \quad \text{and} \quad K = \sum_{j \in J} |k_j|$$

Since Q, C, K do not depend on S , finding a subset S maximizing $\Gamma^{c,q}$ is equivalent to finding a minimum cut separating the additional vertices 0 and $n+1$. Since w is positive, this problem is solvable in polynomial time, using the Gomory and Hu algorithm [9], as it will be explained in the next section.

7 Experimental results

The experiments are conducted on a single thread on a machine with Intel(R) Xeon(R) CPU E5-2630 v2 @2.60GHz, and 16Gb RAM. We use the solver Cplex version 12.6.3.0, and the open source C++ optimization library LEMON [8]. The branching scheme and the management of the current bounds is done by Cplex.

- *Implementation of the separation algorithm*

The separation of inequalities (Q1) and (Q2) is implemented using the so-called Callback functions proposed by Cplex. The separation algorithm consists in:

1. Computing the weights $w_{\{i,j\}}$ introduced in Section 6 according to the value of variables e, t, δ, x (resp. e', t', δ, x) in the solution provided by Cplex
2. Running the Gomory and Hu algorithm [9] provided by LEMON to obtain the Gomory-Hu tree rooted in 0
3. Finding all minimum cost edges along the path between 0 and $n+1$ in the Gomory-Hu tree
4. Testing for any of such edges if the related cut W/\overline{W} such that $0 \in W$ corresponds to a negative value
5. Adding in the model the inequality (Q1) (resp. (Q2)) associated to S , where $S = W \setminus \{0\}$, if there exists.

- *Benchmark*

We test our three formulations on the benchmark proposed by Biskup and Feldmann [5], available online on OR-Library [4]. For each number of tasks $n \in \{10, 20, 50\}$, ten triples (p, α, β) of $(\mathbb{N}_*^n)^3$ are given. For each one, setting $d = hp(J)$ for $h \in \{0.2, 0.4, 0.6, 0.8, 1\}$, gives five instances, including one with an unrestrictive due date corresponding to $h = 1$. We obtain 15-task (resp. 30-task or 40-task) instances, by considering only the first tasks in the 20-task (resp. 50-task) instances. In the following, the average values considered are computed over the ten instances proposed by this benchmark for fixed value of n and h , unless otherwise specified.

In [23], Sourd succeeded to solve instances of this benchmark having up to 1000 tasks. The running time does not exceed 1400 seconds, and the average running time for 1000-tasks instances is between 611 and 918 seconds depending on the value of h . He obtained these results thanks to a dedicated branch-and-bound algorithm using Lagrangian relaxation and dynamic programming. Conversely, our approach is generic for this class of problem and could be more easily extended to problems including additional constraints.

- *Entries of the following tables*

n : the number of tasks

h : the parameter setting the due date d to $hp(J)$ (in the general case only)

#opt : number of instances optimally solved among the ten proposed by the benchmark

avg-T : the average running time in seconds over the optimally solved instances

min-T : the minimum running time in seconds over the optimally solved instances

max-T : the maximum running time in seconds over the optimally solved instances

gap : the average gap over the instances not solved to optimality, that is the relative gap between the best lower and upper bounds

#nd : the average number of nodes over the instances (resp. not) solved to optimality

7.1 Formulations for the unrestrictive case

In this section the problem is solved using formulations (F1) and (F2), as well as the formulation (F_{LO}) proposed in [5] based on linear-ordering variables, for sake of comparison.

- *Formulation (F_{LO})*

Table 1 presents the results obtained with the formulation (F_{LO}). Here we use default Cplex (without Callback functions and with all possible features). This configuration will be denoted as Default configuration. The time limit is set to 3600 seconds.

n	#opt	Optimally solved instances				Not opt. solved instances	
		avg-T	min-T	max-T	#nd	gap	#nd
10	10	9	5	23	11230	-	-
15	5	1821	255	3002	2762970	10%	5315709
20	0	-	-	-	-	144%	1268881

Table 1: Solving the unrestrictive case using (F_{LO})

• *Formulation (F1)*

Table 2 presents the results obtained with the formulation (F1). In addition to the integer variables δ , we define x variables as integer variables to allow Cplex to add more cuts. We use every Cplex feature which is not disabled by the Callback functions used to implement the separation of inequalities (Q1) and (Q2). This configuration will be denoted as Feature+ configuration. The time limit is set to 3600 seconds, except for the 50-task instances, for which the time limit is set to 10000 seconds.

n	#opt	Optimally solved instances				Not opt. solved instances	
		avg-T	min-T	max-T	#nd	gap	#nd
10	10	< 1	< 1	< 1	4	-	-
20	10	5	4	5	105	-	-
30	10	50	34	96	690	-	-
40	10	477	226	706	3871	-	-
50	9	4721	2626	9771	15375	2%	33450

Table 2: Solving the unrestrictive case using (F1)

• *Formulation (F2)*

Table 3 presents the results obtained with the formulation (F2). Here only δ variables are defined as integer variables, and we turn off all Cplex features, since a preliminary test has shown that results are much better doing this. For instance, for $n = 50$, the average running time is 1315 seconds with Cplex features, and 215 seconds without using them. Note that in this configuration, branching is necessarily only done on δ variables.

The time limit is set to 3600 seconds, except for the 60-task instances, for which the time limit is set to 10000 seconds. With a time limit of 3600 seconds, only five over the ten 60-task instances are solved to optimality, and the average gap over the five others is 6%.

n	#opt	Optimally solved instances				Not opt. solved instances	
		avg-T	min-T	max-T	#nd	gap	#nd
10	10	4	3	5	10	-	-
20	10	3	3	4	197	-	-
30	10	5	3	7	1380	-	-
40	10	22	11	43	8545	-	-
50	10	215	69	466	53210	-	-
60	10	4063	262	9543	455873	-	-

Table 3: Solving the unrestrictive case using (F2)

Solving 50-task instances takes in average 215 seconds using (F2), while formulation (F1) achieves to solve 40-task instances within 8 minutes in average and formulation (F_{LO}) does not allow to solve any 20-task instance within one hour. To conclude, for the unrestrictive instances (F2) is the best formulation and allows to solve quite efficiently up to 60-task instances. Moreover, the most efficient configuration for this formulation is to turn off Cplex features.

7.2 Formulations for the general case

In this section the problem is solved using formulations (F3) and (F_{LO}). The time limit is set to 3600 seconds for both tables.

- *Formulation (F_{LO})*

Table 4 presents the results obtained with the formulation (F_{LO}) using the Default configuration, for instances with $h \in \{0.2, 0.4, 0.6, 0.8\}$. The results for $h=1$ are already presented in Table 1.

n	h	#opt	Optimally solved instances				Not opt. solved instances	
			avg-T	min-T	max-T	#nd	gap	#nd
10	0.2	10	14	6	23	30790	-	-
	0.4	10	18	5	19	36011	-	-
	0.6	10	10	4	18	12645	-	-
	0.8	10	8	4	17	10015	-	-
15	0.2	0	-	-	-	-	73%	3372585
	0.4	0	-	-	-	-	59%	3562377
	0.6	5	2278	1422	3105	2722027	24%	4452452
	0.8	5	1575	338	2281	2308746	9%	5438115
20	0.2	0	-	-	-	-	437%	1162593
	0.4	0	-	-	-	-	245%	1278346
	0.6	0	-	-	-	-	159%	1324055
	0.8	0	-	-	-	-	145%	1302895

Table 4: Solving the general case using (F_{LO})

- *Formulation (F3)*

Table 5 presents the results obtained with the formulation (F3) using the Feature+ configuration.

We can observe that using (F3), instances with $h = 0.2$ are easier to solve than those with a larger h . It could be explained by a lower number of candidate partitions (E, T) , since the time between 0 and d is reduced. For the general case, formulation (F3) is better than (F_{LO}). Indeed, solving up to 20-task instances takes less than 3 minutes using (F3), while using (F_{LO}) does not allow to solve all 15-task instances within one hour.

Note that for the unrestrictive instances (with $h = 1$), formulation (F3) is less efficient than (F2), and even than (F1). The following paragraph will exploit this remark.

- *What is really an unrestrictive instance?*

We have defined a due date as unrestrictive as soon as $d \geq \sum p_j$, since it is the common definition. But according to Biskup and Feldmann [5], a due date must be said unrestrictive if solving the problem for an arbitrary due date gives a solution for this due date.

This definition raises two issues. First, since for some instances there exist several optimal solutions whose early tasks do not have the same total length, this definition depends on the algorithm, or even on the

n	h	#opt	Optimally solved instances				Not opt. solved instances	
			avg-T	min-T	max-T	#nd	gap	#nd
10	0.2	10	< 1	< 1	< 1	107	-	-
	0.4	10	< 1	< 1	< 1	278	-	-
	0.6	10	1	< 1	2	289	-	-
	0.8	10	1	< 1	2	272	-	-
	1.0	10	2	< 1	10	253	-	-
20	0.2	10	25	17	33	2919	-	-
	0.4	10	73	28	145	8360	-	-
	0.6	10	86	38	159	8957	-	-
	0.8	10	74	28	144	7662	-	-
	1.0	10	77	27	136	7851	-	-
30	0.2	10	966	107	2142	35085	-	-
	0.4	3	1582	381	1514	62017	7%	83820
	0.6	7	1750	392	3494	56221	15%	44827
	0.8	5	1120	444	2929	46641	10%	74973
	1.0	6	1383	517	3372	48450	10%	71672

Table 5: Solving the general case using (F3)

execution of the algorithm. Secondly, this definition requires an optimal solution to be found to say if the instance is unrestrictive or not. Therefore the prior definition is more convenient. But this remark leads to the following (F2)-(F3) procedure to solve a general instance:

1. Solving the instance without considering d using the formulation (F2)
2. Testing if the total duration of early tasks is smaller than the due date *i.e.* $\sum_{\delta_j=1} p_j \leq d$.

If it is the case, then the solution obtained is optimal.

Otherwise solving the instance considering d using (F3)

In average, the total length of the early tasks in the optimal solutions is 60% of the total length. That means that in this benchmark, instances with $h \geq 0.6$ (*i.e.* $d \geq 0.6p(J)$) are mostly unrestrictive as defined by Biskup and Feldmann [5]. For these instances, the (F2)-(F3) procedure can be relevant (but we do not present corresponding numerical results).

7.3 Comparison of the linear relaxations in the unrestrictive case

Table 6 shows that the lower bound provided by the linear relaxation (F2-LP) of (F2) is far from the optimal value (see the third column). Note that other experiments show that (F1) provides the same lower bound. We try to strengthen this lower bound by adding Cplex cuts and/or the triangle inequalities introduced in [17].

For $n \leq 30$, adding the Cplex cuts provides a better lower bound than adding the triangle inequalities, and combining both of them does not provide a better lower bound. Conversely, for $n \geq 40$, adding the triangle inequalities provides a much better lower bound than adding the Cplex cuts, and combining both of them provides almost the same bound as adding only triangle inequalities, but reduces the running time. For instance, for 60-task instances, adding triangle inequalities reduces the gap from 92.8% to 23.5%, and combining them with the Cplex cuts reduces the running time from 746 seconds to 337 seconds.

These observations lead to look for other valid inequalities for the quadratic polytope defined and studied by Padberg in [17], in order to strengthen our formulations. Indeed, as triangle inequalities, such inequalities

n	(F2-LP)		(F2-LP) +Cplex Cuts		(F2-LP) +Triangle		(F2-LP) +Triangle +Cplex Cuts	
	time	gap	time	gap	time	gap	time	gap
10	0.14	41.1%	2.72	0.00%	0.05	3.29%	1.61	0.00%
20	0.03	67.9%	3.19	0.00%	0.52	13.2%	2.11	10.3%
30	0.12	77.0%	4.86	3.72%	0.52	19.4%	11.7	18.1%
40	0.29	82.9%	9.86	26.7%	31.9	21.5%	48.3	20.9%
50	0.62	86.1%	26.6	42.1%	177	22.5%	145	22.4%
60	0.74	92.8%	375	44.9%	746	23.5%	337	23.5%

Table 6: Improvement of the lower bound by adding Cplex cuts and triangle inequalities

can improve the lower bounds given by the linear relaxation of (F2), but also (F1) and (F3), where δ and x variables satisfy the same inequalities. These observations also drive to deal with the related algorithmic aspects. Indeed, since directly adding such inequalities in the model increases significantly the computation times, we should define how to manage these inequalities, for instance by providing a cutting-plane based algorithm.

8 Conclusion

In this paper, we proposed three new formulations based on earliness/tardiness variables to solve the common due date scheduling problem. Our formulations allow to solve unrestricted instances with up to 50 tasks and general instances up to 20 tasks within few minutes, while Sourd [23] presents a dedicated algorithm solving general instances up to 1000 tasks in less than 15 minutes. Even if our results are far from those presented in [23], our MIP formulations outperform the compact linear ordering variable based MIP proposed in [5]. Our aim for the present work was to provide a MIP formulation less dedicated than the methods used in [23], in order to easily include additional constraints. A key part in our work is the theoretical study of the non overlapping inequalities, in particular Lemmas 3 and 4 and the scheme of proof used for Theorems 12 and 13 (resp. 22 and 23), which should allow to extend our approach to other related scheduling problems.

Further works will focus on the earliness-tardiness scheduling problem with parallel machines, where each machine imposes the same due date for all the tasks. Another issue is to address the single machine common due date scheduling problem with machine unavailability constraints. For both problems formulations similar to (F3) can be derived.

An interesting issue is to study the polyhedra associated to such formulations, to strengthen them using facet defining inequalities, as triangle inequalities, which can be used in any formulation using δ and x variables to describe a cut in a graph.

References

- [1] Kenneth R. Baker and Gary D. Scudder. Sequencing with earliness and tardiness penalties: A review. *Operations Research*, 38(1):22–36, 1990.
- [2] Egon Balas. On the facial structure of scheduling polyhedra. *Mathematical Programming*, Vol 24:179–218, 1985.
- [3] Francisco Barahona and Ali Ridha Mahjoub. On the cut polytope. *Math. Program.*, 36(2):157–173, 1986.

- [4] Dirk Biskup and Martin Feldmann. Common due date scheduling. <http://people.brunel.ac.uk/~mas-tjbb/jeb/orlib/schinfo.html>.
- [5] Dirk Biskup and Martin Feldmann. Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. *Computers & OR*, 28(8):787–801, 2001.
- [6] José R. Correa and Andreas S. Schulz. Single-machine scheduling with precedence constraints. *Math. Oper. Res.*, 30(4):1005–1021, 2005.
- [7] Martin E. Dyer and Laurence A. Wolsey. Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics*, 26(2-3):255–270, 1990.
- [8] LEMON – Library for Efficient Modeling and Optimization in Networks.
- [9] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.
- [10] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.
- [11] Nicholas G. Hall, Wieslaw Kubiak, and Suresh P. Sethi. Earliness-tardiness scheduling problems, II: deviation of completion times about a restrictive common due date. *Operations Research*, 39(5):847–856, 1991.
- [12] Nicholas G. Hall and Marc E. Posner. Earliness-tardiness scheduling problems, I: weighted deviation of completion times about a common due date. *Operations Research*, 39(5):836–846, 1991.
- [13] J.A. Hoogeveen and S.L. van de Velde. Scheduling around a small common due date. *European Journal of Operational Research*, 55(2):237 – 242, 1991.
- [14] John J. Kanet. Minimizing the average deviation of job completion times about a common due date. *Naval research logistics quarterly*, Vol 28:643–651, Dec 1981.
- [15] John J. Kanet and V. Sridharan. Scheduling with inserted idle time: Problem taxonomy and literature review. *Operations Research*, 48(1):99–110, 2000.
- [16] Arthur Kramer and Anand Subramanian. A unified heuristic and an annotated bibliography for a large class of earliness-tardiness scheduling problems. *Journal of Scheduling*, online, 2017.
- [17] Manfred Padberg. The boolean quadric polytope: Some characteristics, facets and relatives. *Math. Program.*, 45(1-3):139–172, 1989.
- [18] Jean-Claude Picard and H. Donald Ratliff. Minimum cuts and related problems. *Networks*, 5(4):357–370, 1975.
- [19] Maurice Queyranne. Structure of a simple scheduling polyhedron. *Mathematical Programming*, 58:263–285, 1993.
- [20] Maurice Queyranne and Andreas S. Schulz. Polyhedral approaches to machine scheduling. Technical Report 408, TU Berlin, 1994, revised 1996.
- [21] Maurice Queyranne and Yaoguang Wang. Single-machine scheduling polyhedra with precedence constraints. *Math. Oper. Res.*, 16(1):1–20, 1991.
- [22] Wayne E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66, 1956.
- [23] Francis Sourd. New exact algorithms for one-machine earliness-tardiness scheduling. *INFORMS Journal on Computing*, 21(1):167–175, 2009.
- [24] Shunji Tanaka and Mituhiko Araki. An exact algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times. *Computers & OR*, 40(1):344–352, 2013.