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

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

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.

Keywords: Just-in-time scheduling, Mixed integer programming formulation, polyhedral approaches

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 $p_j \geqslant 1$).

A schedule can be then encoded by the vector of its completion times $(C_j)_{j\in J}$. Such an encoding allows us 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 [20] 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 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 unit 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 follows.

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

When $d \geqslant \sum p_j$, the common due date is called *unrestrictive* since the due date does not restrict the total duration of early tasks [13]. In this case, the so-called V-shaped dominance property [13] 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 [13], there exists an optimal schedule without idle time and with an *on-time* task, *i.e.* completing exactly at d. For

the common due date setting, idle time only refers to an idle time between tasks, regardless of the interval between 0 and the starting time of the schedule. The problem with an unrestrictive common due date is NP-hard even if $\alpha_j = \beta_j$ for any task $j \in J$ [13]. However, if $\alpha_j = \beta_j = 1$ for any task $j \in J$, the problem is solvable in polynomial time [15].

In the general case, there might be a *straddling* task, *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 $j \in J$ [14, 12].

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 solved by several approaches: with heuristics (*e.g.* [5], [17]), with branch-and-bound algorithms (*e.g.* [24]), and with dynamic programming methods (*e.g.* [14], [25]). The reader can refer to the seminal surveys of [1], [17] and [16] for the early results in this field.

Furthermore, there exist several ways to encode a single machine schedule leading to distinct formulations. Such encodings can be based on completion times, time-indexed variables, linear ordering, positional date and assignement variables [21]. Some of these encodings allow to formulate just-in-time scheduling problems as Mixed Integer Program (MIP). However, few solving approaches based on these formulations 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 task non-overlapping, *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. [2] and [20] propose seminal works in this research line. 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.* [8]) or precedence constraints (*e.g.* [7], [22]).

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 infeasible schedule. In this context, providing a linear formulation describing this polyhedron is not sufficient. [20] 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, we provide MIP based methods to solve a core problem in just-in-time scheduling. Such approaches can be easily extended to tackle other variants embedding this core structure, in contrast with the dedicated methods commonly used in scheduling field. We use natural variables to handle the common due date problem, dealing with a non-regular criterion. Using few additional 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. Thanks to these theoretical results, 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 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 by [5] and also used by [24], as well as a new benchmark covering larger processing times. For sake of comparison, MIP formulations of the literature are also considered.

This article is organized as follows. Section 2 presents basic tools to express the task non-overlapping. We recall Queyranne's linear inequalities for the non-overlapping [20]. We also provide 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.

$$\text{positivity} \qquad \forall j \in J,\ y_j \geqslant p_j \tag{0}$$

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

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, [20] introduces linear inequalities using completion times to handle the non-overlapping. We first recall notations and results proposed by [20] 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 $y \in \mathbb{R}^J$,

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

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

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

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

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

$$\forall S \subseteq J,\ p{*}y\,(S) \geqslant g_p(S) \tag{Q0}$$

We denote by 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 $p_j y_j \geqslant p_j^2$, that is $y_j \geqslant p_j$ since $p_j > 0$. So constraints (0) ensure that the inequalities (Q0) associated to the singletons are all satisfied. If $|S| \geqslant 2$, 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 $\forall (i,j) \in J^2,\ i \prec j \Rightarrow y_j \geqslant y_i + p_j$. Using inequalities (0) we deduce that $y_j \geqslant p(I) + p_j$ 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 $k \geqslant 1$ and let $S \subseteq J$ with $|S| = k+1$. By setting $j = \max_{\prec} S$ and $U = S \backslash \{j\}$, then, on one hand, by induction $p{*}y(U) \geqslant g_p(U)$, and, on the other, by previous arguments $y_j \geqslant p(U) + p_j$. Consequently $p{*}y(S) = p{*}y(U) + p_j y_j \geqslant g_p(U) + p_j\big(p(U) + p_j\big) = g_p(S)$ using (3), hence y satisfies the inequality (Q0) associated to S. □

Some points in $\text{conv}(Q)$ correspond to infeasible 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 overlap. 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, [20] 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 $\text{conv}(Q)$. [20] 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 (*[20]*)

(i) $extr(P^Q) \subseteq Q$
(ii) $P^Q = conv(Q)$

Moreover, [20] 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 [23], 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 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 overlap 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 $y_i \leqslant y_j < y_i + p_j$,*
***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).*

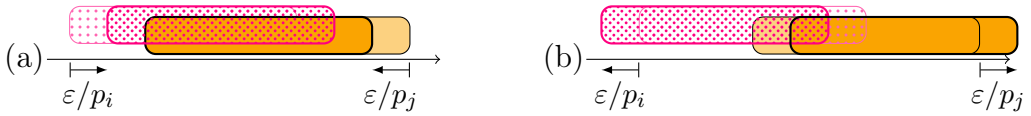

Figure 1: Illustration of the schedules disruption between y and y^{+-} (a) (resp. y^{-+} (b))

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), $m_1 \geqslant 0$ and $m_2 \geqslant 0$, thus $\varepsilon \geqslant 0$.
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 $p * y^{+-}(S) = p * y(S) \geqslant g_p(S)$.
If $i \in S$ and $j \in S$ then $p * y^{+-}(S) = p * y(S) + p_i\frac{\varepsilon}{p_i} - p_j\frac{\varepsilon}{p_j} = p * y(S) \geqslant g_p(S)$.
If $i \notin S$ and $j \in S$ then $p * y^{+-}(S) = p * y(S) - p_j\frac{\varepsilon}{p_j} \geqslant g_p(S)$ since $\varepsilon \leqslant m_1$.
If $i \in S$ and $j \notin S$ then $p * y^{+-}(S) = p * y(S) + p_i\frac{\varepsilon}{p_i} \geqslant p * y(S) \geqslant g_p(S)$.

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

Claim
Let $(i,j) \in J^2$. **If** $y_i \leqslant y_j$, **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 $U = S \backslash \{j\}$, 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 $p*y(U) \geqslant g_p(U)$ from inequalities (Q0), we deduce that $p_j y_j \leqslant p_j p(S)$ and even $y_j \leqslant p(S)$ since $p_j > 0$.
Moreover $p*y\big(S \sqcup \{i\}\big) = p*y(S) + p_i y_i = g_p(S) + p_i y_i \leqslant g_p(S) + p_i y_j$ by assumption.
Using these two inequalities, we get $p*y\big(S \sqcup \{i\}\big) \leqslant g_p(S) + p_i p(S) < g_p(S) + p_i\big[p(S) + p_i\big]$ since $p_i > 0$. Furthermore, $g_p(S) + p_i\big[p(S) + p_i\big] = g_p\big(S \sqcup \{i\}\big)$ from (3) and $g_p\big(S \sqcup \{i\}\big) \leqslant p*y\big(S \sqcup \{i\}\big)$ from inequality (Q0). We finally get $p*y\big(S \sqcup \{i\}\big) < p*y\big(S \sqcup \{i\}\big)$, 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 $y_i \leqslant p(S)$.
Moreover $p*y\big(S \sqcup \{j\}\big) = p*y(S) + p_j y_j = g_p(S) + p_j y_j < g_p(S) + p_j\big[y_i + p_j\big]$ by assumption.
Using these two inequalities, we can write $p*y(S \sqcup \{j\}) < g_p(S) + p_j\big[p(S) + p_j\big]$ since $p_j > 0$.
Furthermore, $g_p(S) + p_j\big[p(S) + p_j\big] = g_p\big(S \sqcup \{j\}\big)$ from (3) and $g_p\big(S \sqcup \{j\}\big) \leqslant p*y\big(S \sqcup \{j\}\big)$ from inequality (Q0). We finally get $p*y\big(S \sqcup \{j\}\big) < p*y(S \sqcup \{j\})$, 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 overlap, 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 overlap, 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 schedules can be adapted to a more complex polyhedron, that is a polyhedron defined by inequalities (Q0) and additional inequalities. Indeed, it is then sufficient to check that the two vectors C^{+-} and C^{-+} also satisfy 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 $M \geqslant p(J)$), the additional inequalities are the following.

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

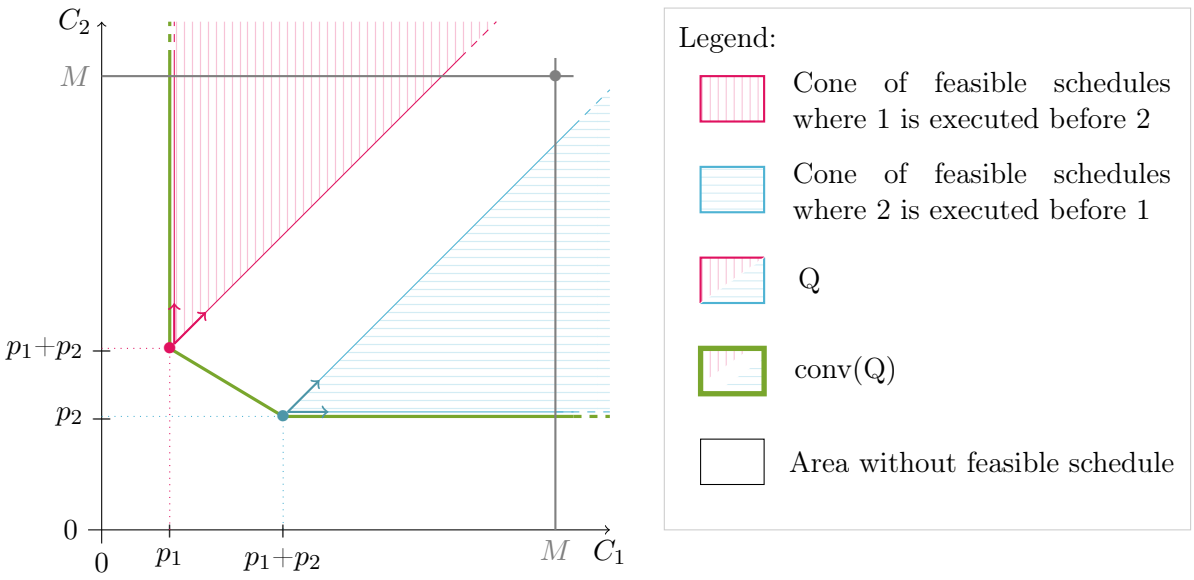

Figure 2: Q and $\text{conv}(Q)$ in the case of two tasks

Note that inequalities (4) induce extreme points encoding infeasible schedules as depicted in Figure 2 for a 2-task instance. Adding the inequalities $C_1 \leqslant M$ and $C_2 \leqslant M$ leads to the extreme point (M, M) which encodes a schedule with an overlap. 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 non-negatively weighted sum of variables. For any given polyhedron P of $\mathbb{R}^n$, we consider the following set of extreme points which are unique minimizer of such function. The unicity is required to deal with some zero weights.

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

Since the extreme points are exactly the points that can be written as the unique minimizer of a linear function, $\text{extr}^*(P) \subseteq \text{extr}(P)$. Let $P^{Q,M}$ denote the polytope defined by inequalities (Q0) and (4). Let us assume that $C \in P^{Q,M}$ is the completion time vector of a schedule with an overlap. If one of the two

overlapping tasks has a completion time equal to M, applying Lemma 3 to C provides a vector C^{-+} which does not satisfy inequalities (4) and therefore is not in $P^{Q,M}$. Point y cannot be proved to not be extreme in $P^{Q,M}$. In order to prove that such point is not a unique minimizer, we provide the following lemma.

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 $y_j \geqslant p(J)$,*
***then** there exists $\varepsilon\in\mathbb{R}_+^*$ such that $y-\frac{\varepsilon}{p_j}\mathbb{1}_j$ also satisfies 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 $y-\frac{\varepsilon}{p_j}\mathbb{1}_j$ also satisfies inequalities (Q0) and that $\varepsilon\geqslant 0$. It remains to show that $\varepsilon>0$, that is 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.

$$p{*}y\,(S) > g_p(S) \Leftrightarrow p{*}y\,(U) + p_j y_j > g_p(U) + p_j\big[p(U)+p_j\big] \Leftrightarrow p{*}y\,(U) - g_p(U) > p_j\big[p(S)-y_j\big]$$

If $S\subsetneq J$, then $p(S)<p(J)\leqslant y_j$, thus $p_j\big[p(S)-y_j\big]<0$. Moreover $p{*}y\,(U)-g_p(U)\geqslant 0$ 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 $p_j\big[p(S)-y_j\big]\leqslant 0$ since $y_j\geqslant p(J)$. In this case, $p_j\big[p(S)-y_j\big]$ can be equal to zero if $y_j=p(J)$, but we prove that $p{*}y\,(U)-g_p(U)>0$ as follows.

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

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

Combining Lemmas 3 and 4, we prove that a vector C in $\mathrm{extr}^*(P^{Q,M})$ is in Q, that is it encodes a feasible schedule by its completion times. Indeed, since such a vector C satisfies inequalities (Q0), an overlap between tasks i and j such that $C_i\leqslant C_j<C_i+p_j$ 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 $C_j\geqslant p(J)$, 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 has a smaller value than C for any linear function with positive (or zero) coefficients, 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 every left-tight schedule is encoded by an extreme point of $P^{Q,M}$, and even by a vector of $\mathrm{extr}^*(P^{Q,M})$.

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.* $d \geqslant p(J)$. 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 [13]. 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 $\theta(C) = \Big(\big([d-C_j]^+\big)_{j\in J}, \big([C_j-d]^+\big)_{j\in J} \Big)$. Using function θ, we have $f_{\alpha,\beta} = g_{\alpha,\beta} \circ \theta$.

- *Consistency between e and t using δ variables*

A vector (e,t) in $(\mathbb{R}_+)^J \times (\mathbb{R}_+)^J$ is *consistent* if $\forall j \in J$, either ($e_j \geqslant 0$ and $t_j = 0$) or ($e_j = 0$ and $t_j \geqslant 0$). There exists C in $\mathbb{R}^J$ such that $\theta(C) = (e,t)$ if and only (e,t) is consistent. In order to ensure consistency, we introduce the following inequalities using new boolean variables $(\delta_j)_{j\in J}$. For each task j, δ_j indicates if j is early.

$$\forall j \in J,\ e_j \geqslant 0 \qquad (5)$$
$$\forall j \in J,\ e_j \leqslant \delta_j\,(p(J) - p_j) \qquad (6)$$
$$\forall j \in J,\ t_j \geqslant 0 \qquad (7)$$
$$\forall j \in J,\ t_j \leqslant (1-\delta_j)\,p(J) \qquad (8)$$

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 on 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 $d - p_j$. of 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 on the tardiness in a d-block. Consequently, we have the following lemma.

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* $C = \big(d - e_j + t_j\big)_{j \in J}$ *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 $d - e_j + t_j \geqslant d - e_j \geqslant d - p(J) + p_j$. Since d is unrestrictive, we deduce that $d - e_j + t_j \geqslant p_j$. Hence, we obtain the following lemma.

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 do not overlap each other, and that tardy tasks are fully processed after d and 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.

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

For a tardy task, the tardiness can be seen as a completion time with respect to d. Therefore, ensuring that the tardy tasks are fully processed before d (resp. they do not overlap each other) is equivalent to imposing positivity constraints for tardy tasks (resp. the non-overlapping constraint for tardy tasks). As shown on Figure 3, for an early task j, the value $e_j + p_j$ can be seen as a completion time. Using $x_{/S}$ to denote $\big(x_j\big)_{j \in S}$ for any subset S of J and for any vector x in $\mathbb{R}^J$, the following lemma sums up these observations.

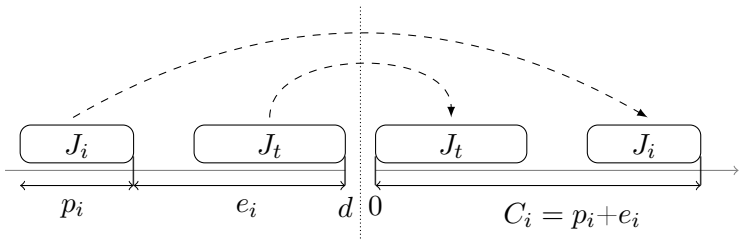

Figure 3: Illustration of the role of $p_i + e_i$ for an early task i

Lemma 8

Let $C \in \mathbb{R}^J$ *and set* $(e, t) = \theta(C)$. ***If*** *there is no* $j \in J$ *such that* $C_j - p_j < d$ *and* $C_j > d$,
then C *satisfies (1)* $\Leftrightarrow$ $(e+p)_{/E(C)}$ *and* $t_{/T(C)}$ *satisfy (0) and (1).*

In the formulation, δ describes the partition between early and tardy tasks, denoted as follows.

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

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.

$$\forall S \subseteq J, \qquad p * (e+p)\big(S \cap E(\delta)\big) \geqslant g_p(S) \tag{9}$$

$$\forall S \subseteq J, \qquad p * t\big(S \cap T(\delta)\big) \geqslant g_p(S) \tag{10}$$

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

$p_j t_j \geqslant p_i^2 + p_j^2 + p_i p_j$ 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 by δ_j (resp. by $(1-\delta_j)$). If $\delta_j \in \{0,1\}$, then $(1-\delta_j)^2 = (1-\delta_j)$, $e_j\,\delta_j = e_j$ from inequality (6) and $t_j\,(1-\delta_j) = t_j$ from inequality (8). We obtain the following quadratic inequalities.

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

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

• *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.

$$\forall (i,j) \in J^<, \qquad x_{i,j} \geqslant \delta_i - \delta_j \tag{13}$$

$$\forall (i,j) \in J^<, \qquad x_{i,j} \geqslant \delta_j - \delta_i \tag{14}$$

$$\forall (i,j) \in J^<, \qquad x_{i,j} \leqslant \delta_i + \delta_j \tag{15}$$

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

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

Lemma 9 (*[9]*)

If $\delta \in \{0,1\}^J$ ***then*** *for all* $(i,j) \in J^<$*:*
(i) δ *and* x *satisfy* (13)-(16) *associated with* (i,j) $\Leftrightarrow$ $x_{i,j} = 0$ *if* $\delta_i = \delta_j$ *and* $x_{i,j} = 1$ *otherwise.*
(ii) In case (i) holds, then $\delta_i \delta_j = \dfrac{\delta_i + \delta_j - x_{i,j}}{2}$ *and* $(1-\delta_i)(1-\delta_j) = \dfrac{2 - (\delta_i + \delta_j) - x_{i,j}}{2}$.

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.

$$\forall S \subseteq J, \ \sum_{j \in S} p_j e_j \geqslant \sum_{(i,j) \in S^<} p_i p_j \,\frac{\delta_i + \delta_j - x_{i,j}}{2} \tag{Q1}$$

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

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

Lemma 10

Let $(\delta, x) \in \{0,1\}^J \times \mathbb{R}^{J^<}$ *satisfying inequalities* (13)-(16).
(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* $S \subseteq J \Leftrightarrow (e+p)_{/E(\delta)}$ *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* $S \subseteq J \Leftrightarrow t_{/T(\delta)}$ *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*** $E(\delta)=\mathsf{E}\big(\theta^{-1}(e,t)\big)$ *and* $T(\delta)=\mathsf{T}\big(\theta^{-1}(e,t)\big)$.

Proof: Let $C=\theta^{-1}(e,t)$. If $j\in\mathsf{T}(C)$, then $C_j>d$ by definition. That is $t_j>e_j$ since $C_j=d-e_j+t_j$. From inequality (5), we deduce that $t_j>0$ and from inequality (8), that $\delta_j\neq 1$. Since δ_j is an integer, $\delta_j=0$. That proves $\mathsf{T}(C)\subseteq T(\delta)$. 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 $t_j\geqslant p_j>0$ from inequality (Q2) for $S=\{j\}$, we deduce that $C_j>d$, that proves $T(\delta)\subseteq\mathsf{T}(C)$.
Similarly, we can prove the equality for the early tasks (without using (Q1)). □

• *Formulation (F1)*

Let us define $\boldsymbol{P^1}=\big\{\ (e,t,\delta,x)\in\mathbb{R}^J\times\mathbb{R}^J\times[0,1]^J\times\mathbb{R}^{J^<}\ \big|\ (5)\text{-}(8),\ (13)\text{-}(16),\ (Q1)\text{ and }(Q2)\text{ are satisfied}\ \big\}$. Note that this polyhedron does not depend on either α, β, or even d, but is only defined 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 an 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 $\mathbb{R}^J\times\mathbb{R}^J\times\mathbb{R}^J\times\mathbb{R}^{J^<}$, $\mathrm{int}_\delta(V)=\big\{(e,t,\delta,x)\in V\,|\,\delta\in\{0,1\}^J\big\}$. 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 from 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 the following.

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

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 any straddling task such that tasks are processed between $d-p(J)$ and $d+p(J)$, i.e. $\forall j\in J,\ d-p(J)\leqslant C_j-p_j$ and $C_j\leqslant d+p(J)$*
then *there exists $X=(e,t,\delta,x)\in int_\delta(P^1)$, such that $\theta(C)=(e,t)$.*

Proof: From C, let us set: $(e,t)=\theta(C)$, $\delta=\mathbb{1}_{E(C)}$, $x=\big(\mathbb{1}_{\delta_i\neq\delta_j}\big)_{(i,j)\in J^<}$ 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 $E(\delta)=\mathsf{E}(C)$ (resp. $T(\delta)=\mathsf{T}(C)$), 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 $\forall j\in J,\ d-p(J)\leqslant C_j-p_j$ (resp. $C_j\leqslant d+p(J)$) 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 to $\mathrm{int}_\delta(P^1)$ since $\delta\in\{0,1\}^J$. □

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

Theorem 13

Let $X^* = (e, t, \delta, x) \in int_\delta(P^1)$.
If $X^* \in extr(P^1)$ *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, (e, t) 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 by $\mathcal{S}^*$. 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, $E(\delta) = \mathsf{E}(C^*)$ (resp. $T(\delta) = \mathsf{T}(C^*)$), 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 $e_i + p_i \leqslant e_j + p_j < (e_i + p_i) + p_j$. Two cases have to be considered:

$\rightarrow$ If $e_j + p_j < p(J)$, then from Lemma 3 on $(e+p)_{/E}$ there exists $\varepsilon \in \mathbb{R}_+^*$ such that setting $e^{+-} = e + \frac{\varepsilon}{p_i}\mathbb{1}_i - \frac{\varepsilon}{p_j}\mathbb{1}_j$ and $e^{-+} = e - \frac{\varepsilon}{p_i}\mathbb{1}_i + \frac{\varepsilon}{p_j}\mathbb{1}_j$, both $(e^{+-}+p)_{/E}$ and $(e^{-+}+p)_{/E}$ satisfy (Q0). Using Lemma 10, both e^{+-}, δ, x and e^{-+}, δ, x satisfy (Q1). Since changing the value of ε for $\min\big(\varepsilon,\, p(J) - p_j - e_j\big)$ does not affect the satisfaction of (Q1), we can assume $\varepsilon \leqslant p(J) - p_j - e_j$, while ensuring $\varepsilon > 0$. Since $e_i^{+-} = e_i + \frac{\varepsilon}{p_i} \leqslant e_i + \varepsilon$, using this latter assumption and $e_j + p_j \geqslant e_i + p_i$, we obtain $e_i^{+-} \leqslant p(J) - p_i$. For k in $J \setminus \{i\}$, $e_k^{+-} \leqslant e_k$, and since e satisfies (6), we deduce that $e_k^{+-} \leqslant p(J) - p_k$. Thus e^{+-} satisfies inequalities (6).

Besides, since $(e^{+-}+p)_{/E}$ satisfies inequalities (Q0) for the singletons, $e_k^{+-} + p_k \geqslant p_k$ 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.

$\rightarrow$ If $e_j + p_j \geqslant p(J)$, then $e_j + p_j \geqslant p(E)$, and from Lemma 4 on $(e+p)_{/E}$ 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^-, δ, x satisfy (Q1). Since e^- is component-wise smaller than e, e^- also satisfies inequalities (6). Besides, the inequality (Q0) for the singleton $\{j\}$ ensures that $e_j^- \geqslant 0$, 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 $g_{\alpha,\beta}$.

Finally, $(e+p)_{/E}$ 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 between tasks and, if there is no early task, shifting backward all the tasks such that the first one becomes on-time. Since the due date is unrestrictive, no task is scheduled before 0 despite the backward shifting, then $\widehat{\mathcal{S}}$ is a d-block by construction. If $\widehat{C}$ denotes the completion times defining $\widehat{\mathcal{S}}$, then $\forall j \in J$, $d - p(J) \leqslant \widehat{C}_j - p_j$ and $\widehat{C}_j \leqslant d + p(J)$. Then using Theorem 12, there exists $\widehat{X} = (\hat{e}, \hat{t}, \hat{\delta}, \hat{x}) \in \text{int}_\delta(P^1)$, such that $\theta(\widehat{C}) = (\hat{e}, \hat{t})$. 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 tardiness.

Then $g_{\alpha,\beta}(\hat{e}, \hat{t}) = f_{\alpha,\beta}(\widehat{C}) < f_{\alpha,\beta}(C^*) = g_{\alpha,\beta}(e, t)$, 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

(i) Any optimal d-block is encoded by a vector minimizing $g_{\alpha,\beta}$ on $\text{int}_\delta\big(\text{extr}(P^1)\big)$.
(ii) Conversely, any vector minimizing $g_{\alpha,\beta}$ on $\text{int}_\delta\big(\text{extr}(P^1)\big)$ 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 $P^{\delta^*}=\big\{(e,t)\,|\,(e,t,\delta^*,x^*)\in P^1\big\}$, which is the slice of P^1 according to δ^*, *i.e.* the projection of set of points of P^1 satisfying $\delta=\delta^*$ and therefore $x=x^*$.
To show that X^* is 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 $(e^*,t^*)=\frac{1}{2}\big((e^1,t^1)+(e^2,t^2)\big)$, and (e^*,t^*) would not be an extreme point of P^{δ^*}.

Let (E,T) denote the partition of tasks given by δ^*, *i.e.* $E=E(\delta^*)$ and $T=T(\delta^*)$. Using Lemma 10, we decompose P^{δ^*} as a Cartesian products of polyhedra as follows.

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

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, where the index set J is replaced by T while keeping $M=p(J)\geqslant p(T)$. Similarly, $P^{\delta^*,E}$ is a translation according to $-p_{/E}$ of $P^{Q,M}$, where J is replaced by E while keeping $M=p(J)\geqslant p(E)$. 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 $\text{int}_\delta\big(\text{extr}(P^1)\big)$.

To prove item *(i)*, it remains to show that X^*, or more precisely (e^*,t^*), is a minimizer of $g_{\alpha,\beta}$. By contradiction, let us assume that there exists $\widehat{X}=(\hat{e},\hat{t},\hat{\delta},\hat{x})\in\text{int}_\delta\big(\text{extr}(P^1)\big)$ such that $(\hat{e},\hat{t})$ minimizes $g_{\alpha,\beta}$ and $g_{\alpha,\beta}(\hat{e},\hat{t})<g_{\alpha,\beta}(e^*,t^*)$. According to Theorem 13, $\widehat{X}$ encodes a schedule inducing a total penalty $g_{\alpha,\beta}(\hat{e},\hat{t})$, which is lower than the total penalty of $\mathcal{S}^*$ a contradiction.

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

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).

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

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 with 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 [11].

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 $X = (e,t,\delta,x) \in \text{extr}\big(P^1 \cap F^{J_0,J_1}\big)$.

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 $X \in \text{extr}\big(P^1 \cap F^{T(\delta),E(\delta)}\big)$. Since $P^1 \cap F^{T(\delta),E(\delta)}$ is exactly the set denoted by 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 $X \in \text{extr}(P^1)$. □

Note that in general, such an algorithm $\mathcal{A}$ is not sufficient to minimize a linear function under both integrity and extremality constraints in a polyhedron. To illustrate this observation, let us consider the following formulation. where int_y denotes the operator keeping only the points (y,z) such that y is an integer.

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

$\mathcal{A}$ 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 polyhedra are respectively $(1, 2+\frac{2}{3})$, and $(2,2)$. The provided point is then $(1, 2+\frac{2}{3})$, with a value of $2+\frac{2}{3}$ whereas the best value for an integer extreme point is 2, reached by $(0,2)$.

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

For any formulation (F), let us denote by 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 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 $X^* = (e^*,t^*,\delta,x) \in \text{int}_\delta\big(\text{extr}(P^1)\big)$, we get $g_{\alpha,\beta}(e^*,t^*) \geqslant \text{value(F1)} \geqslant \text{value(F1-int)} = g_{\alpha,\beta}(e,t)$.

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 problem associated with (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 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 [14], 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 $\frac{\alpha_j}{p_j}$ (resp. nonincreasing $\frac{\beta_j}{p_j}$) 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 are scheduled in nondecreasing order of α_j/p_j and the tardy ones in nonincreasing order of 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 Lemmas 5 and 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 from $[\![1,n]\!]$ to J such that:

$$\left(\frac{\alpha_{\rho(k)}}{p_{\rho(k)}}\right)_{k\in[\![1,n]\!]} \text{ and } \left(\frac{\beta_{\sigma(k)}}{p_{\sigma(k)}}\right)_{k\in[\![1,n]\!]} \text{ are nonincreasing.}$$

We say that a feasible schedule is *ρ-σ-shaped* when early (resp. tardy) tasks are processed in decreasing order of ρ^{-1} (resp. increasing order of σ^{-1}). These schedules are dominant in the unrestrictive case, and will only be considered in 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:

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

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

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

where we use $x_{i,j}$ without carrying if $i<j$, that is to denote the variable $x_{\min(i,j),\max(i,j)}$.

Therefore, the total penalty is simply expressed by $h^{\rho,\sigma}_{\alpha,\beta}(\delta,x) = f_{\alpha,\beta}\big(\, e^{\rho}(\delta,x),\, t^{\sigma}(\delta,x)\,\big)$, which is linear. We then consider the polyhedron $\boldsymbol{P^2} = \big\{\ (\delta,x) \in [0,1]^J \times \mathbb{R}^{J^<} \ \big|\ \text{(13)-(16) are satisfied}\ \big\}$.
By definition of $e^{\rho}(\delta,x)$ and $t^{\sigma}(\delta,x)$, a vector (δ,x) in P^2 cannot encode an infeasible 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

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

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 [18]. 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 positive or zero unit earliness penalties. We can no longer derive an optimal solution from the one obtained for the instance which does not include zero unit earliness penalty 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 $J = [\![1, 3]\!]$, $d = 6$, $p_1 = 5$, $p_2 = 3$, $p_3 = 2$, $\alpha_1 = 0$, $\beta_1 = 1$ and $\alpha_2 = \beta_2 = \alpha_3 = \beta_3 = 2$, then the optimal schedule is given by $C_2 = 4 \leqslant d$, $C_3 = 6 = d$, $C_1 = 11 \geqslant d$. 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 [12].

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: $J = [1..8]$, $\forall i \in [1..6]$, $p_i = 1$, $\alpha_i = 40$, $\beta_i = 4$, $p_7 = 3$, $\alpha_7 = 20$, $\beta_7 = 8$, $p_8 = 4$, $\alpha_8 = 20$, $\beta_8 = 11$, and $d = 2$. We can easily verify that the optimal partition is $E = \emptyset$, $T = J$ and $a = 2$. 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.

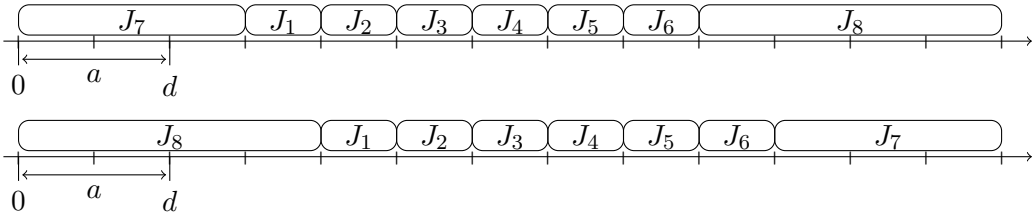

Figure 4: The two types of dominant schedules for $E = \emptyset$ and $T = J$.

The best ones are those starting by task 7 and ending by task 8. Nevertheless the ratio $\beta_7/p_7 = 8/3$ is smaller than the ratio $\beta_8/p_8 = 11/14$. 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: $J = [\![1, 5]\!]$, $\forall i \in [1..3]$, $p_i = 1$, $\alpha_i = 10$, $\beta_i = 2$, $p_4 = 4$, $\alpha_4 = 10$, $\beta_4 = 5$, $p_5 = 3$, $\alpha_5 = 10$, $\beta_5 = 3$ and $d = 2$ We can easily verify that the optimal partition is

$E=\emptyset$, $T=J$ and $a=2$. 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.

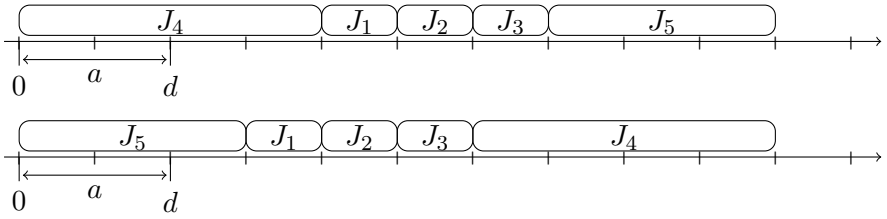

Figure 5: The two types of dominant schedules for $E=\emptyset$ and $T=J$.

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 $p(E)\leqslant d$), 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 lost 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 $p_j\geqslant d-p(E)$. 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. 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.

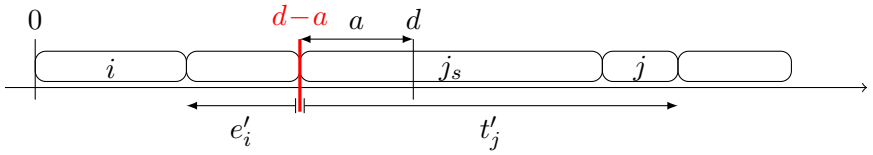

Figure 6: The (a,e',t') encoding for a schedule holding a straddling task j_s

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. Unfortunately, to ensure that a takes the expected value in case of a

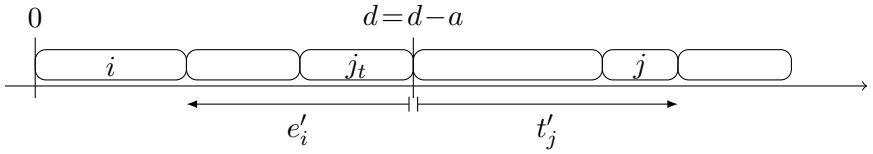

Figure 7: The first (a, e', t') encoding for a schedule holding an on-time task j_t

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 also used for a schedule holding an on-time task and having tardy tasks, as illustrated by Figure 8.

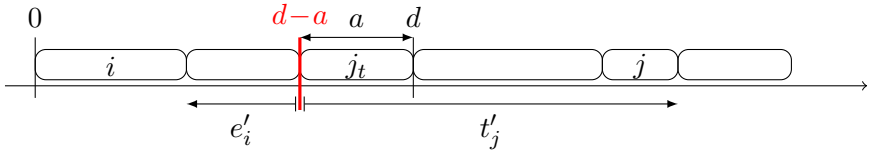

Figure 8: The second (a, e', t') encoding for a schedule holding an on-time task j_t

To sum up, the first encoding, with $a=0$, is suitable for d-blocks, except those without tardy tasks, and 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 : $C = \left(d-a-e'_j+t'_j\right)_{j\in J}$ 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. 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 by (5')-(8') in the sequel.

Note that δ_j no longer necessarily indicates if task j is early or not. Keeping the previous notations $E(\delta) = \{j \in J \mid \delta_j = 1\}$ and $T(\delta) = \{j \in J \mid \delta_j = 0\}$, $\big(E(\delta), T(\delta)\big)$ 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 $\widetilde{E}(C) = \{\, j \in J \mid C_j < d\}$ and $\widetilde{T}(C) = \{\, j \in J \mid C_j \geqslant d\,\}$.

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

• *Handling the positivity*

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

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

• *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 by (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 $C_j \leqslant M$ is compatible with the non-overlapping inequalities (Q0) only if $M \geqslant p(J)$. 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, let us consider the instance defined by $J=[\![1,2]\!]$, $d=5$, $p_1=p_2=3$, $\alpha_1=\alpha_2=1$, $\beta_1=\beta_2=10$, and the polyhedron $P=\left\{ (e,t,\delta,x,a)\in\mathbb{R}^J\times\mathbb{R}^J\times[0,1]^J\times\mathbb{R}^{J^<}\times\mathbb{R} \;\middle|\; \text{(5')-(8')}, \text{(13)-(16)}, \text{(Q1')},\text{(Q2')}, \text{(17)} \right\}$ defined by the inequalities introduced for the general case so far. The vector $X=(2,2,0,0,1,1,0,0)$, is an integer extreme point of P. It corresponds to the schedule $\mathcal{S}$ where both tasks complete at time 3, since $e'_1=e'_2=2$ and $a=0$. The induced penalty is 4, which is the minimal penalty over P. However, $\mathcal{S}$ is infeasible since the two tasks overlap. This overlapp occurs in spite of inequalities (Q1') because $d-a=5<6=p(E)$, implying that $d-a$ is a too restrictive upper bound in inequalities (17). To prevent this restrictiveness, we introduce the following inequality.

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

To ensure that inequalities (Q1') (resp. (Q2')) prevent overlaps of tasks completing before (resp. after) $d-a$ do not overlap, the total penalty must be a nonincreasing function of variable e'_j (resp. t'_j) for each task j such that $\delta_j=1$ (resp. $\delta_j=0$). We have to provide linear inequalities ensuring that the variable a takes a value such that the objective function fulfils these two conditions. 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, then these two conditions are ensured.

• *Ensuring that a takes the expected value*

In spite of their apparent symmetry, the two conditions 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 with $\{j\}$, *i.e.* $e'_j\geqslant 0$, task j remains early and its tardiness decreases, which reduces the induced penalty. Therefore, the first constraint is guaranteed by the following inequality.

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

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 with $\{j\}$, *i.e.* $t'_j\geqslant p_j$, 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.

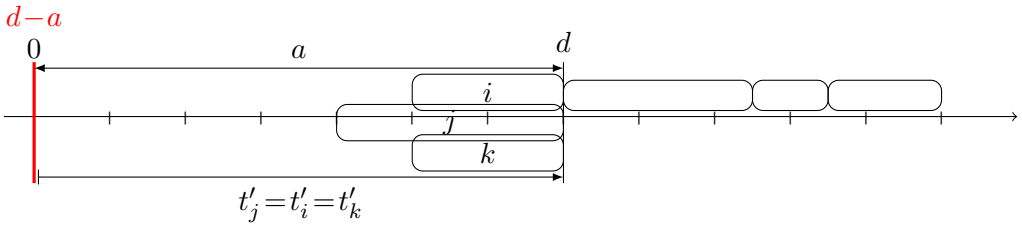

Figure 9: An infeasible schedule when $a=d$

Note that this case appears even if we add inequalities $\forall j\in J,\ t'_j\geqslant a(1-\delta_j)$. Adding inequalities $\forall j\in J,\ p_j\geqslant a(1-\delta_j)$, 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.

$$\sum_{j\in J}\gamma_j = 1 \tag{20}$$

$$\forall j\in J,\qquad \delta_j \leqslant 1-\gamma_j \tag{21}$$

$$\forall j\in J,\qquad t'_j \leqslant p_j + (1-\gamma_j)\,(p(J)-p_j) \tag{22}$$

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 $t'_{i_0}\leqslant p_{i_0}$,

and since $t'_{i_0} \geqslant p_{i_0}$ 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,δ) satisfies (20)-(21) $\Leftrightarrow \exists i_0 \in T(\delta)$, $\gamma = \mathbb{1}_{i_0}$
*(ii) **If** (i) holds and t', δ, x satisfy (13)-(16), (22) and (Q2'), **then** $t'_{i_0} = p_{i_0}$ and $\forall j \in T(\delta)$, $j \neq i_0$, $t'_j \geqslant t'_{i_0} + p_j$.*

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

$$\forall j \in J, \qquad a \leqslant p_j + (1-\gamma_j)\, d \tag{23}$$

• *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-overlap 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.

$$\forall j \in J, \qquad b_j \geqslant 0 \tag{24}$$
$$\forall j \in J, \qquad b_j \leqslant a \tag{25}$$
$$\forall j \in J, \qquad b_j \leqslant \delta_j d \tag{26}$$
$$\forall j \in J, \qquad b_j \geqslant a - (1-\delta_j)\, d \tag{27}$$

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 the following.

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

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

Note that if the schedule holds a straddling task, then $\theta(C) = \widetilde{\theta}'(C)$, since $\mathsf{E}(C) = \widetilde{\mathsf{E}}(C)$ and $\mathsf{T}(C) = \widetilde{\mathsf{T}}(C)$. 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 $h_{\alpha,\beta}\big(\theta'(C)\big) = h_{\alpha,\beta}\big(\widetilde{\theta}'(C)\big) = f_{\alpha,\beta}\big(\theta'(C)\big)$.

• *Formulation (F3)*

Let us define the polyhedron

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

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. We introduce the operator $int_{\delta,\gamma}$, which only keeps points with integer δ and γ. For $V \subseteq \mathbb{R}^J \times \mathbb{R}^J \times \mathbb{R}^J \times \mathbb{R}^{J^<} \times \mathbb{R} \times \mathbb{R}^J \times \mathbb{R}^J$, the set $\text{int}_{\delta,\gamma}(V) = \left\{(e', t', \delta, x, a, b, \gamma) \in V \mid \delta \in \{0,1\}^J,\ \gamma \in \{0,1\}^J\right\}$. Finally, our formulation for the general common due date problem is

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

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 $(e', t', a, b) = \theta'(C) = \widetilde{\theta}'(C)$,*
then *C satisfies (1) $\Leftrightarrow$ $(e'+p)_{/E(C)}$ and $t'_{/T(C)}$ satisfies (0) and (1).*

(ii) ***If*** *there exists $j_t \in J$ such that $C_{j_t} = d$ and $(e', t', a, b) = \widetilde{\theta}'(C)$,*
then *C satisfies (1) $\Leftrightarrow$ $(e'+p)_{/E(C)}$ and $t'_{/T(C)}$ satisfies (0) and (1).*

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

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

Proof: Let us start by proving (i).

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

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

Since task j_s starts at or after 0 and before d, *i.e.* $0 \leqslant C_{j_s} - p_{j_s} < d$, we have $0 < a \leqslant d$. Thus inequality (19) is satisfied, and for any task $j \neq j_s$, $a \leqslant d + p_j = (1-\gamma_j)\,d + p_j$. More precisely, task j_s starts after all early tasks, and since they do not overlap, $p(E) \leqslant C_{j_s} - p_{j_s} = d - a$, 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, $C_j \leqslant C_{j_s} - p_{j_s} = d - a$ 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 $p_j + (1-\gamma_j)\big(p(J) - p_j\big) = p(J) \geqslant 0$. By assumption $C_j \geqslant d - p(J) + p_j$, thus $e'_j = d - a - C_j \leqslant p(J) - p_j$, 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 $C_j - p_j \geqslant 0$, thus $d - e'_j - p_j\delta_j \geqslant a$ and inequality (17) is satisfied for j.

For a task j in T, $C_j \geqslant d \geqslant d-a$, then $e'_j = 0$ and $t'_j = C_j - (d-a)$. The corresponding inequality (6') is thus satisfied. Moreover, $d - e'_j - p_j\delta_j = d \geqslant a$, then inequality (17) is satisfied for j. By assumption $C_j \leqslant C_{j_0} - p_{j_0} + p(J) \leqslant d + p(J)$, thus $t'_j \leqslant \big(d+p(J)\big) - (d-a)$ and then $t'_j \leqslant p(J)$. We deduce that the corresponding inequality (8') is also satisfied, as well as inequality (22), since $p_j + (1-\gamma_j)\big(p(J)-p_j\big)$ 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 $\mathrm{int}_{\delta,\gamma}(P^3)$.

Rewriting the proof by replacing θ' by $\widetilde{\theta'}$, $E(C)$ by $\widetilde{E}(C)$, $T(C)$ by $\widetilde{T}(C)$, 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 $a = p_{j_s} + (d - C_{j_s}) = p_{j_s} = p_{j_s} + (1-\gamma_{j_s})\, d$. □

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

Theorem 23

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

If some tasks have a zero unit 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 unit earliness penalty, an overlap would appear at time 0.
Since their unit 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 unit 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* $int_{\delta,\gamma}\big(extr(P^3)\big)$.
Conversely, any vector minimizing $h_{\alpha,\beta}$ *on* $int_{\delta,\gamma}\big(extr(P^3)\big)$, *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 $\forall S\subseteq J,\ \Gamma^{c,q}(S)=\sum\limits_{(i,j)\in S^<}q_{i,j}+\sum\limits_{i\in S}c_i$.

Indeed we have: X satisfies $(Q1)\Leftrightarrow\forall S\subseteq J,\ \sum\limits_{(i,j)\in S^<}p_ip_j\,\dfrac{\delta_i+\delta_j-x_{i,j}}{2}\leqslant\sum\limits_{i\in S}p_ie_i$

$$\Leftrightarrow\forall S\subseteq J,\ \sum_{(i,j)\in S^<}p_ip_j\,(\delta_i+\delta_j-x_{i,j})-2\sum_{i\in S}p_ie_i\leqslant 0$$

$$\Leftrightarrow\forall S\subseteq J,\ \Gamma^{c^1,q^1}(S)\leqslant 0.$$

where $c^1=-2\left(p_je_j\right)_{j\in J}$ and $q^1=\left(p_ip_j\,(\delta_i+\delta_j-x_{i,j})\right)_{(i,j)\in J^<}$. 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 $c^2=2\left((1-\delta_j)p_j^2-p_jt_j\right)_{j\in J}$ and $q^2=\left(p_ip_j\,(2-(\delta_i+\delta_j)-x_{i,j})\right)_{(i,j)\in J^<}$.

Note that in both definitions of Γ^{c^1,q^1} and Γ^{c^2,q^2}, the parameter q is non-negative 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 [19]. Let us assume that $J=[\![1,n]\!]$ for sake of brevity. We consider the weighted undirected graph $G=(V,A,w)$, where $V=[\![0,n+1]\!]$, $A=\big\{\{i,j\}\,|\,(i,j)\in V^2,\{i,j\}\neq\{0,n+1\}\big\}$, $\forall j\in J,\ w_{\{0,j\}}=[k_j]^+$, $w_{\{j,n+1\}}=[k_j]^-$ where $k_j=2c_i+\sum\limits_{i=1}^{j-1}q_{i,j}+\sum\limits_{k=j+1}^{n}q_{j,k}$, and $\forall(i,j)\in J^<,\ w_{\{i,j\}}=q_{i,j}$. Figure 10 gives an illustration of such a graph for $n=5$.

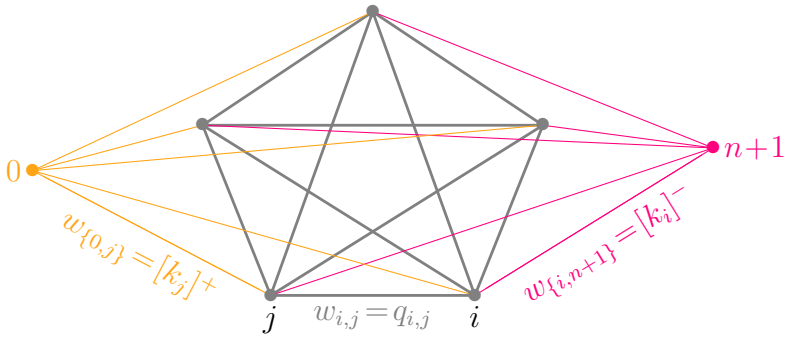

Figure 10: Illustration of the weighted undirected graph G for $n=5$

Note that V and A 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.* $w(W,\overline{W})=\sum\limits_{i\in W,j\in\overline{W}}w_{\{i,j\}}$.

Let us introduce three constants: $Q=\sum\limits_{(i,j)\in J^<}q_{i,j}$, $C=\sum\limits_{j\in J}c_j$ and $K=\sum\limits_{j\in J}|k_j|$.

Hence, for any $S\subseteq J$: $$\Gamma^{c,q}(S)=-\frac{1}{2}\,w\big(S\cup\{0\},[\![1,n+1]\!]\backslash S\big)+\frac{Q+C}{2}+\frac{K}{4}.$$

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 [10] algorithm, 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 [6]. The branching scheme and the management of the current bounds is done by CPLEX. The time limit is set to 3600 seconds. For sake of comparison, all the formulations use CPLEX Default.

- *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 the following steps.

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 [10] algorithm 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.

Due to these Callback functions, some CPLEX features are disabled in (F1) and (F3).

- *Biskup and Feldmann's benchmark*

We test our three formulations on the benchmark proposed by [5], available online on OR-Library [4]. For each number of tasks $n \in \{10, 20, 50\}$, ten triples (p, α, β) of $\left(\mathbb{N}_*^n\right)^3$ are given. For each one, setting $d = \lfloor h\, p(J) \rfloor$ 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 30-task and 40-task instances, by considering only the first tasks of 50-task instances. In the following, the average values considered are computed over the ten instances proposed by this benchmark for fixed values of n and h, unless otherwise specified.

[24] succeeded in solving 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. However, Sourd's approach is based on a time-indexed formulation which involves $O(np(J))$ variables and hence nodes in the graph used for computing the Lagrangian lower bound. The Biskup and Feldmann's benchmark considers small values for the job processing times which ensures a fast computation time of the Lagrangian lower bound.

- *New benchmark with long processing times*

In the Biskup and Feldmann's benchmark, processing times range is $[1, 20]$. We propose a benchmark where processing times are randomly drawn from the uniform distribution $\mathcal{U}\left[\frac{p_{max}}{10}, p_{max}\right]$ for $p_{max} \in \{100, 200, 300\}$. For each $p_{max} \in \{100, 200, 300\}$ and each $n \in \{10, 20, 30, 40, 50\}$, we randomly generate ten triples (p, α, β) of $\left(\mathbb{N}_*^n\right)^3$. For each task j, α_j and β_j are randomly drawn from the uniform distribution $\mathcal{U}[1..20]$. By setting $d = \lfloor h\, p(J) \rfloor$ for $h \in \{0.2, 0.4, 0.6, 0.8, 1\}$, each triple gives five instances, including one unrestrictive, which results in 750 instances.

- *MIP formulations from the literature*

In order to assess our formulation efficiency, we implement two other MIP formulations proposed in the literature: the formulation ($\mathrm{F_{LO}}$) based on linear-ordering variables proposed by [5] and the time-indexed formulation ($\mathrm{F_{TI}}$) used in [24].

- *Entries of the following tables*

  n : the number of tasks

  h : the parameter setting the due date d to $\lfloor h\,p(J) \rfloor$ (in the general case only)

  #opt : number of instances optimally solved among the ten proposed by the benchmark under the 3600 seconds time limit

  avg-T : the average 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

7.1 Formulations for the unrestrictive case

In this section the problem is solved using formulations (F1) and (F2), as well as formulations (F_{LO}) and (F_{TI}). Table 1 presents the results obtained on Biskup and Feldmann's benchmark, while Table 2 presents those obtained on long processing times instances, for $p_{max} \in \{100, 200, 300\}$.

n	(F_{LO})			(F_{TI})			(F1)			(F2)		
	#opt	avg-T	gap	#opt	avg-T	gap	#opt	avg-T	gap	#opt	avg-T	gap
10	10	9	-	10	1	-	10	0	-	10	3	-
20	0	-	144%	10	4	-	10	2	-	10	3	-
30				10	15	-	10	44	-	10	7	-
40				10	40	-	10	637	-	10	106	-
50				10	41	-	1	1388	16%	10	1315	-

Table 1: Solving Biskup and Feldmann's unrestrictive instances using (F_{LO}), (F_{TI}), (F1) and (F2)

As shown in Table 1, (F_{LO}) is unable to solve any 20-task instance within the time limit. Thus, (F_{LO}) is not used neither for larger instance size, nor for the new benchmark. Other experiments show that (F_{LO}) can only solve 5 over 10 instances for $n = 15$. (F_{TI}) is able to optimally solve Biskup and Feldmann's instances up to size 50 in less than 40 seconds. (F1) is able to optimally solve instances up to size 30 in around 40 seconds. In contrast, ten minutes are required to optimally solve 40-task instances and (F1) fails to solve 50-task instances within the time limit. However, other experiments show that, under a time limit of 10 000 seconds, (F1) solves 9 over the 10 instances for $n = 50$, with an average computation time of 4721 seconds. (F2) is able to optimally solve all the instances up to size 50 within the time limit. Other experiments conducted without CPLEX features show that (F2) can be faster: 22 seconds for $n = 40$, 215 seconds for $n = 50$ and 4063 seconds for $n = 60$.

As shown in Table 2, the efficiency of (F_{TI}) greatly depends on the value of parameter p_{max}, which was expected since the number of variables is related to the length of the horizon, *i.e.* $2\,p(J)$. While it only takes 40 seconds in average to solve all the 50-task Biskup and Feldmann's instances, (F_{TI}) solves 8 over the 10 50-task instances in the new benchmark for $p_{max} = 100$, within 690 seconds in average. In addition, (F_{TI}) fails at solving any instance for $p_{max} = 300$ due to memory limitations. CPLEX could not even provide a solution or a lower bound in this case. We can notice that for 20-task instances, the computation time required is at least 360 seconds for $p_{max} = 200$ and $p_{max} = 300$. (F1) is able to optimally solve all instances up to size 30 regardless of p_{max} value, faster that (F_{TI}). The same observation holds for (F2) up to size 40 regardless of p_{max} value.

To sum up for the unrestrictive case, (F_{TI}) gives the bests results for the Biskup and Feldmann's benchmark. However, this formulation is sensitive to the total length of the processing times (*i.e.* $p(J)$), and is unable to solve the 50-task instances with long processing times ($p_{max} = 300$). In contrast, the results obtained with (F1) and (F2) do not significantly get worse with processing time increase.

p_{max}	n	(F_{TI})			(F1)			(F2)		
		#opt	avg-T	gap	#opt	avg-T	gap	#opt	avg-T	gap
100	10	10	6	-	10	0	-	10	3	-
	20	10	74	-	10	3	-	10	3	-
	30	10	186	-	10	68	-	10	13	-
	40	10	494	-	8	1335	4%	10	294	-
	50	8	690	0%	0	-	22%	9	1743	2%
200	10	10	15	-	10	0	-	10	3	-
	20	10	361	-	10	3	-	10	3	-
	30	10	886	-	10	56	-	10	12	-
	40	7	1322	0%	6	1173	8%	10	359	-
	50	7	1289	2%	1	1859	29%	4	1738	6%
300	10	10	27	-	10	0	-	10	3	-
	20	10	380	-	10	4	-	10	3	-
	30	6	1508	3%	10	87	-	10	15	-
	40	8	2533	0%	9	1572	11%	10	210	-
	50	x	x	x	0	-	29%	5	2662	5%

Table 2: Solving unrestrictive instances generated with p_{max} 100, 200, 300 using (F_{TI}), (F1) and (F2)

7.2 Formulations for the general case

In this section the problem is solved using formulations (F3) as well as (F_{LO}) and (F_{TI}). Table 3 presents the results obtained on the Biskup and Feldmann's benchmark, while Table 4 presents those obtained on long processing times instances, for $p_{max} = 200$.

n	h	(F_{LO})			(F_{TI})			(F3)		
		#opt	avg-T	gap	#opt	avg-T	gap	#opt	avg-T	gap
10	0.2	10	1	-	10	1	-	10	0	-
	0.4	10	1	-	10	1	-	10	1	-
	0.6	10	1	-	10	1	-	10	1	-
	0.8	10	1	-	10	1	-	10	1	-
20	0.2	0	-	437%	10	3	-	10	36	-
	0.4	0	-	245%	10	4	-	10	116	-
	0.6	0	-	159%	10	4	-	10	125	-
	0.8	0	-	145%	10	3	-	10	118	-
30	0.2				10	17	-	10	1255	-
	0.4				10	22	-	3	1620	6%
	0.6				10	9	-	4	962	8%
	0.8				10	13	-	5	1405	9%

Table 3: Solving Biskup and Feldmann's restrictive instances using (F_{LO}), (F_{TI}) and (F3)

As shown in Table 3, (F_{LO}) is unable to solve any restrictive instance for $n = 20$ within the time limit.

Thus, (F_{LO}) is not used neither for larger instance size, nor for the new benchmark. Other experiments show that (F_{LO}) cannot solve 15-task instance when $h=0.2$ and $h=0.4$. When $h=0.6$ (resp. $h=0.8$), (F_{LO}) solves 5 over the 10 instances for $n=15$, using in average 2278 seconds (resp. 1575 seconds). (F_{TI}) is able to optimally solve all the Biskup and Feldmann's restrictive instances. Moreover, the computation times are similar to those obtained for the unrestrictive instances: less than 25 seconds for 30-task instances. (F3) is able to optimally solve all the instances up to size 20 as well as the 30-task instances when $h=0.2$. However, the computation time is much larger than for (F_{TI}): around 2 minutes for $n=20$ and 20 minutes for $n=30$ when $h=0.2$.

p_{max}	n	h	(F_{TI})			(F3)		
			#opt	avg-T	gap	#opt	avg-T	gap
200	10	0.2	10	22	-	10	1	-
		0.4	10	24	-	10	1	-
		0.6	10	15	-	10	1	-
		0.8	10	14	-	10	1	-
200	20	0.2	10	116	-	10	30	-
		0.4	9	343	1%	10	91	-
		0.6	10	299	-	10	93	-
		0.8	10	333	-	10	89	-
200	30	0.2	8	821	2%	10	1377	-
		0.4	7	1293	3%	4	1143	4%
		0.6	10	803	-	7	1479	5%
		0.8	10	740	-	7	1166	6%

Table 4: Solving restrictive instances generated with $p_{max}=200$ using (F_{TI}) and (F3)

As shown in Table 4, for long processing time instances with $p_{max}=200$, (F_{TI}) optimally solves almost all the instances within the time limit up to $n=20$. It is important to notice that, for similar size, (F_{TI}) computation time is significantly larger for long processing times instances than for Biskup and Feldmann's ones: for $n=20$, at least 116 seconds against a few seconds. In contrast, (F3) optimally solves all instances up to size 20 along with 30-task instances when $h=0.2$. Note that, for these instances, (F3) is rather faster than (F_{TI}) for $n=20$. However, (F3) fails to solve instances with $n=30$.

For general case instances, we obtain the same conclusion drawn for the unrestrictive instances. (F_{TI}) is faster than (F3) for Biskup and Feldmann's instances, while this is not the case for long processing times instances.

Other experiments show that (F3) used on unrestrictive Biskup and Feldmann's instances (*i.e.* with $h=1$) is less efficient than (F1): 77 seconds in average for the 20-task instances, and more than 1300 seconds for the six optimally solved 30-task instances. The following paragraph will exploit this remark.

- *What is really an unrestrictive instance?*

We have defined a due date as unrestrictive as soon as $d \geqslant \sum p_j$, since it is the common definition. But according to [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 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.* $\sum_{\delta_j=1} p_j \leqslant d$.
   If it is the case, then the solution obtained is optimal.
   Otherwise solving the instance considering d using (F3)

On average on the Biskup and Feldmann's benchmark, 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 $h \geqslant 0.6$ (*i.e.* $d \geqslant 0.6\, p(J)$) are mostly unrestrictive as defined by [5]. For these instances, the (F2)-(F3) procedure can be relevant (but we do not present corresponding numerical results).

7.3 Linear relaxations analysis for (F2)

Table 5 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 by [18].

	(F2-LP)		(F2-LP) + Cplex Cuts		(F2-LP) + Triangle		(F2-LP) + Triangle + Cplex Cuts	
n	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 5: Improvement of the lower bound by adding Cplex cuts and triangle inequalities

For $n \leqslant 30$, 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 $n \geqslant 40$, 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 [18], in order to strengthen our formulations. Indeed, as triangle inequalities, such 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, thanks to our theoretical contributions on the non-overlapping inequalities, we proposed three new formulations based on earliness/tardiness variables to solve the common due date scheduling problem. Our formulations allow to solve unrestrictive instances with up to 50 tasks and general instances up to 20 tasks within few minutes. While scheduling literature proposes pseudo-polynomial methods strongly dependant on the total length of processing times, our formulation size does not rely on this value. Even if our results for the Biskup and Feldmann's benchmark are far from those presented by [24], our MIP formulations outperform the compact MIP formulation based on linear ordering variables. In addition, for instances with long processing times, our formulations outperform the time-indexed formulation in the unrestrictive case. 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*, 24:179–218, 1985.

[3] Francisco Barahona and Ali Ridha Mahjoub. On the cut polytope. *Mathematical Programming*, 36(2):157–173, 1986.

[4] Dirk Biskup and Martin Feldmann. ORLIB common due date scheduling. http://people.brunel.ac.uk/ mastjjb/jeb/orlib/schinfo.html, 1998.

[5] Dirk Biskup and Martin Feldmann. Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. *Computers & Opeartions Research*, 28(8):787–801, 2001.

[6] Coin-OR. LEMON, library for efficient modeling and optimization in networks. http://lemon.cs.elte.hu/, 2003.

[7] José R. Correa and Andreas S. Schulz. Single-machine scheduling with precedence constraints. *Mathematics of Operations Research*, 30(4):1005–1021, 2005.

[8] 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.

[9] R. Fortet. L'algèbre de Boole et ses applications en recherche opérationelle. *Cahiers du Centre d'Études en Recherche Opérationnelle*, 4:5, 1959.

[10] 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.

[11] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[12] 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.

[13] 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.

[14] 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.

[15] John J. Kanet. Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly*, 28:643–651, Dec 1981.

[16] John J. Kanet and V. Sridharan. Scheduling with inserted idle time: Problem taxonomy and literature review. *Operations Research*, 48(1):99–110, 2000.

[17] 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.

[18] Manfred Padberg. The boolean quadric polytope: Some characteristics, facets and relatives. *Mathematical Programming*, 45(1-3):139–172, 1989.

[19] Jean-Claude Picard and H. Donald Ratliff. Minimum cuts and related problems. *Networks*, 5(4):357–370, 1975.

[20] Maurice Queyranne. Structure of a simple scheduling polyhedron. *Mathematical Programming*, 58:263–285, 1993.

[21] Maurice Queyranne and Andreas S. Schulz. Polyhedral approaches to machine scheduling. Technical Report 408, TU Berlin, 1994, revised 1996.

[22] Maurice Queyranne and Yaoguang Wang. Single-machine scheduling polyhedra with precedence constraints. *Mathematics of Operations Research*, 16(1):1–20, 1991.

[23] Wayne E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66, 1956.

[24] Francis Sourd. New exact algorithms for one-machine earliness-tardiness scheduling. *INFORMS Journal on Computing*, 21(1):167–175, 2009.

[25] Shunji Tanaka and Mituhiko Araki. An exact algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times. *Computers & Opeartions Research*, 40(1):344–352, 2013.

Appendix : Proof of Theorem 23

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 $\widetilde{\theta'}$ *i.e.* that $X^*{=}\theta'(C^*)$ or $X^*{=}\widetilde{\theta'}(C^*)$.

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 $\forall j{\in}T(\delta)$, $j{\neq}j_0$, $t'_j \geqslant t'_{j_0}{+}\,p_j$. 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 $T(\delta)$, $C^*_j - p_j = (C^*_{j_0}{-}\,p_{j_0}) + t'j - p_j \geqslant (C^*_{j_0}{-}\,p_{j_0}) + t'_{j_0} = C^*_{j_0} > C^*_{j_0}{-}\,p_{j_0}$. We deduce that $C^*_{j_0}{-}\,p_{j_0} = \min\limits_{j\in T(\delta)} C^*_j - p_j$, and then $a{=}d - \min\limits_{j\in T(\delta)} C^*_j - p_j$.

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

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 $a \leqslant p_{j_0}$, thus $C^*_j > d$. We deduce that $T(\delta)\backslash\{j_0\}{\subseteq}\,T(C^*)$. Conversely, for a task j in $T(C^*)$, $C^*_j > d$, which is equivalent to $t'_j - e'_j > a$. Since $a \geqslant 0$ 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.

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

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

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 $\widetilde{E}(C^*)$, $T(C^*)$ by $\widetilde{T}(C^*)$, and θ' by $\widetilde{\theta'}$ 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 $e' = \left([d{-}a{-}C^*_j]^+\right)_{j\in J}$ and $t' = \left([C^*_j{-}(d{-}a)]^+\right)_{j\in J}$. 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 $p_j \leqslant d{-}a{-}e'_j = C^*_j$. For a task j in T, inequality (17) ensures that $a \leqslant d$, then $C_j = d{-}a + t'_j \geqslant t'_j$. For another, we deduce that $t'_j \geqslant p_j$ 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 $e'_i{+}p_i \leqslant e'_j{+}p_j < (e'_i{+}p_i){+}p_j$. Three cases have to be considered.

$\rightarrow$ If $e'_j{+}p_j \geqslant p(J)$, then $e'_j{+}p_j \geqslant p(E)$. Applying Lemma 4, we can construct a vector e'^- such that $X^- = (e'^-, t', \delta, x, a, b, \gamma)$ is in $\mathrm{int}_{\delta,\gamma}(P^3)$ 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).

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

$\rightarrow$ 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 $\mathrm{int}_{\delta,\gamma}(P^3)$ 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 $t'_i \leqslant t'_j < t'_i{+}p_j$. Since $\forall k{\in}T(\delta)$, $k{\neq}j_0$, $t'_k \geqslant t'_{j_0}{+}\,p_k$, 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:

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

$\rightarrow$ 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 $\widehat{\mathcal{S}}$ by tightening tasks arround d. Using Theorem 22, there exists $\widehat{X} \in \text{int}_{\delta,\gamma}(P^3)$ encoding $\widehat{\mathcal{S}}$, and it contradicts he 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 $a^- = a - \varepsilon$ and $X^- = (e', t', \delta, x, a^-, a^-\delta, \gamma)$ (resp. $a^+ = a + \varepsilon$ and $X^+ = (e', t', \delta, x, a^+, a^+\delta, \gamma)$), 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.