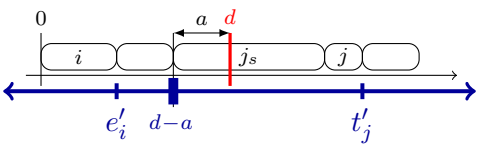
Dominances en programmation linéaire :
ordonnancement autour d'une date d'échéance commune

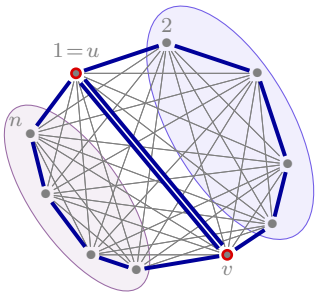
par Anne-Elisabeth Falq

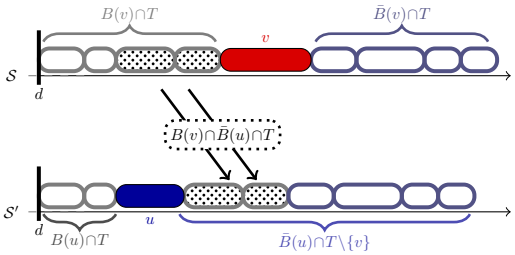
Thèse de doctorat en informatique
dirigée par Pierre Fouilhoux et Safia Kedad-Sidhoum
présentée et soutenue publiquement le 2 Novembre 2020

Jury :

Nadia Brauner	rapporteuse	Professeure, Université Grenoble Alpes, G-SCOP
Pierre Fouilhoux	co-directeur	Professeur, Université Sorbonne Paris Nord, LIPN
Claire Hanen	examinatrice	Professeure, Université de Nanterre, LIP6
Safia Kedad-Sidhoum	co-directrice	Professeure, CNAM, CEDRIC
Quentin Louveaux	examinateur	Professeur associé, Université de Liège, Institut Montefiore
Maurice Queyranne	rapporteur	Professeur émérite, University of British Columbia
Françis Sourd	examinateur	HDR, entreprise Sun'R







Préface

Genèse de la thèse

Cette thèse est certainement née dans le cours MAOA (modèles et algorithmes pour l'ordonnancement et applications) que j'ai suivi en master 2 puisqu'elle résulte de l'envie de mêler les techniques de programmation linéaire que Pierre y présentait aux problématiques d'ordonnancement que Safia y présentait.

Bien qu'elle soit articulée autour du problème d'ordonnancement à une machine avec date d'échéance commune, le but premier de cette thèse n'était pas la résolution de ce problème. L'idée était davantage de voir, d'une part ce que les méthodes polyédrales pouvaient apporter sur un problème d'ordonnancement pour lequel les méthodes de résolution exactes proposées dans la littérature ne relèvent généralement pas de la programmation linéaire, mais plutôt de la programmation dynamique ou du Branch-and-Bound; et d'autre part comment les résultats connus en théorie de l'ordonnancement pouvaient être utilisés pour améliorer les formulations et leur résolution par programmation linéaire.

Organisation et contributions de la thèse

La thèse est découpée en trois parties qui correspondent à trois axes de recherche différents. Elles sont ordonnées suivant l'ordre chronologique des recherches, mais peuvent être lues indépendamment dans la mesure où le matériel commun aux trois parties (définitions, notations, propriétés, exemples...) a été rassemblé en préambule dans le chapitre 0.

Ainsi ce chapitre introductif, en plus de présenter un état de l'art, présente des propriétés de dominance pour les problèmes d'ordonnancement avec date d'échéance commune que nous avons étendues à des instances arbitraires, ainsi qu'une formulation linéaire compacte pour le cas où la date d'échéance est non restrictive, basée sur des variables binaires de partition de l'ensemble des tâches. En fin de chapitre, nous proposons aussi des contre-exemples montrant que des propriétés de dominance vraies pour une date d'échéance non restrictive ne le sont plus si cette date devient restrictive, ce qui nous empêche d'étendre la formulation compacte au cas général.

Dans le but de proposer une formulation pour le cas général, nous nous sommes intéressés dans la partie A à des variables similaires aux variables de date de fin d'exécution de tâche, qu'on appelle variables naturelles. Pour gérer de telles variables dans le cas d'un critère régulier, Maurice Queyranne [34] a proposé des inégalités linéaires dites inégalités de non-chevauchement.

Dans le chapitre 1 nous proposons des formulations mathématiques en variables naturelles basées sur des inégalités de non-chevauchement inspirées de celles de Queyranne, l'une pour le cas non restrictif, l'autre pour le cas général. Au delà de ces deux formulations, le chapitre 1 donne surtout des propriétés sur les inégalités de non-chevauchement et une méthode générique pour construire et prouver des formulations en variables naturelles. Pour preuve de cette généricité, nous proposons en fin de chapitre 1 des ébauches de formulation pour d'autres problèmes d'ordonnancement.

Le chapitre 2 est dédié à la mise en pratique de ces formulations qui ne sont pas exactement des formulations linéaires en nombres entiers : en plus de devoir être des points entiers, les solutions doivent êtres des points extrêmes. On explique à la fois comment gérer cette contrainte d'extrêmalité combinée à la contrainte d'intégrité, et comment séparer les inégalités de non-chevauchement introduites au chapitre 1. Afin de mesurer l'intérêt pratique de ces formulations, nous les avons implémentées avec un solveur de programmation linéaire, ainsi que d'autres formulations linéaires pour comparaison. Les résultats de cette campagne expérimentale – fournis en fin de chapitre 2 – ne sont pas très satisfaisants en comparaison de ceux obtenus par l'algorithme de Branch-and-Bound proposé par Françis Sourd [41].

Malgré cela, dans le cas non restrictif, la formulation compacte proposée au chapitre 0 – qui quant à elle est basée sur des variables de partition – se révèle prometteuse, bien qu'elle offre une valeur de relaxation continue assez faible. Afin de renforcer cette formulation, nous étudions dans la partie B les variables de partition.

Le chapitre 3 présente une revue des principales inégalités proposées pour les différents polyèdres associés au problème MAX-CUT encodé ou non avec des variables de partition. Cette revue est l'occasion de donner un cadre formel pour rassembler ces différentes inégalités sous une même écriture, adaptée à notre formulation compacte. Ces inégalités renforcent une formulation au sens classique du terme : elles coupent des points fractionnaires en vue d'améliorer la valeur de relaxation continue, et ainsi d'accélérer le processus de Branch-and-Bound. On a implémenté certaines d'entre elles pour les ajouter à la formulation compacte et mesurer leur impact. Les résultats numériques de ces expérimentations – présentés en fin de chapitre 3 – ne sont pas probants : si la valeur du relâché continu est considérablement améliorée, le temps de résolution, lui, n'est pas meilleur.

Dans le chapitre 4, on s'intéresse alors à une autre approche de renforcement pour les variables de partition. Remarquant que la partition triviale dans laquelle toutes les tâches sont en avance n'est jamais une solution optimale pour notre problème d'ordonnancement, on s'intéresse non plus au polyèdre des coupes mais à celui des coupes non triviales. Il s'agit d'une première approche d'élimination de solutions non optimales. (c'est-à-dire qu'on coupe deux solutions réalisables, parce qu'on les sait non optimales). Bien que cela revienne seulement à enlever deux points extrêmes, de nombreuses nouvelles facettes apparaissent lorsqu'on passe d'un polyèdre à l'autre. On propose plusieurs familles d'inégalités, et la preuve qu'elles définissent des nouvelles facettes. La quantité, la variété et la taille de ces familles d'inégalités nous ont fait renoncer à les implémenter, et changer légèrement l'approche de renforcement dans la partie suivante.

En effet dans la partie C, on s'intéresse à des inégalités qui, sans être nécessairement facettes, éliminent des points entiers encodant des solutions non optimales, plus précisément des solutions non localement optimales, pour un voisinage donné.

Le chapitre 5 introduit d'abord de telles inégalités, qu'on appelle inégalités de dominance, pour le problème dans le cas non restrictif, avant de proposer un cadre plus général pour ces inégalités. Grâce à ce cadre, une deuxième famille d'inégalités pour le cas non restrictif est rapidement présentée, avant de comparer ces deux familles. En fin de chapitre, on fournit quelques résultats théoriques négatifs à leur propos (elles n'améliorent pas la valeur de relaxation continue).

Le chapitre 6 quant à lui offre des résultats expérimentaux positifs: l'implémentation des inégalités de dominance conduit d'une part à une véritable amélioration de la résolution exacte pour le cas non restrictif avec la formulation compacte, et d'autre part à une procédure heuristique qui fournit rapidement une très bonne borne supérieure.

Dans le chapitre 7, qui présente des travaux en cours, on essaye d'étendre cette approche de renforcement au delà de l'ordonnancement en proposant des inégalités de dominance pour d'autres problèmes d'optimisation combinatoire se formulant comme un programme linéaire en nombre entiers.

Enfin le chapitre 8 propose une conclusion de la thèse, ainsi que des perspectives. Il est suivi par diverses annexes qui ont pour but d'accompagner la lecture (et pas d'être lue à la fin): rappels de définitions usuelles en théorie des graphes et en analyse convexe, tables des notations, liste des inégalités proposées pour l'ordonnancement avec date d'échéance commune, index des formulations et polyèdres, index des définitions.

Guide de lecture

L'introduction se veut assez pédagogique, certaines parties peuvent donc être omises. Par exemple, les personnes déjà familières avec l'ordonnancement juste-à-temps pourront parcourir plus rapidement:

- la section 0.1, *un coup d'œil aux figures peut suffire,*
- la section 0.2, *les énoncés des lemmes 0.1 et 0.2 suffisent*
- la section 0.3, *résumée sur la cartographie page 32.*

De plus, les personnes familères avec les outils de programmation linéaire pourront passer le début de la section 0.4 et ne lire que la sous-section 0.4.5 qui présente quelques formulations linéaires pour l'ordonnancement.

Choix de rédaction

Je précise ici quelques choix de rédaction et de mise en page dont la connaissance pourrait vous faciliter la lecture.

• *Transitions*

Les transitions qui articulent les sections et sous-sections, sont généralement placées à la fin de la section qui précède, et non au début de celle qui suit, pour donner au lecteur un avant goût de ce qui l'attend, si ce n'est de lui donner envie de poursuivre sa lecture. Ainsi, pour une petite introduction quand vous reprenez la lecture, n'hésitez pas à remonter un peu dans le texte, vous retrouverez facilement la transition qui précède car elles ont été écrites en italique.

• *Exemples*

Les exemples et contre-exemples sont souvent présentés sur une page à part et non au fil du texte comme les figures ou les tables. Cela permet de poursuivre facilement la lecture sans s'y attarder, et surtout, cela permet de voir l'exemple en entier sur une page. N'hésitez pas à tourner la page si vous ne voyez pas l'exemple dont il est question.

• *Définitions et notations*

Les définitions des termes utilisés sont données pour la plupart au fil de l'eau. Un index page 219 permet de retrouver à quelle page chaque terme est introduit. Les définitions générales d'analyse convexe et de théorie des graphes font quant à elles l'objet de chapitres proposés en annexe. Les notations sont elles aussi données au cours du texte, et rappelées à partir de la page 213.

• *Numéros de pages*

La numérotation des pages inclut la page de titre et la page de garde. Cela permet que la page numérotée n en bas soit aussi numérotée n par un lecteur PDF, et évite ainsi des pièges à l'impression.

• *Nombre de pages*

Les choix de mise en page ainsi que les nombreuses figures et les annexes rendent ce document volumineux en nombre de pages.

• *Figures et légendes*

J'espère que vous apprécierez les figures qui illustrent ce tapuscrit. Elles ont toutes été réalisées en Tikz (notamment grâce aux explications proposées dans [43]). Les figures sont normalement non ambiguës en noir et blanc, mais plus lisibles en couleurs. Lorsqu'une légende explicite accompagne la figure, elle est présentée dans un double cadre gris (*Cf.* page 14 par exemple), mais une bonne part de la légende est implicite car commune à toutes les figures d'ordonnancement, comme la date d'échéance d tracée en rouge, les tâches en avance en jaune, celles en retard en rose...

Tout ceci étant dit, bonne lecture!

Paris, le 22 Septembre 2020

Contents

C Dominance inequalities

Appendices

Chapter 0

Scheduling around a common due date and related dominance properties

This chapter intends to introduce just-in-time scheduling around a common due date and to present the main known results in this field. In addition, we extend some known properties, provide counter-examples to other property extensions and finally provide two reformulations for a problem.

In Section 0.1, we present the two scheduling problems on which we will focus, as well as the commonly used notations and the representations. For both problems, we give in Section 0.2 some dominance properties and present in Section 0.3 the different known solving approaches to which they lead. Moreover, at the end of Section 0.3 we take a step back and give an overview of related problems. Section 0.4 is dedicated to the different ways to formulate a scheduling problem as a linear program (LP) or as a mixed-integer program (MIP), and finally, Section 0.5 presents a compact MIP for the unrestrictive common due date problem.

> *NB*: All notations introduced in this section and later, as well as commonly used notations, are summarized on page 215 in the notation appendix.

0.1 Scheduling problem definition

In general, a scheduling problem consists in organizing an activity so as to minimize costs or penalties, or so as to maximize profits or utilities (*Cf.* [10], [18], and [33]). Mathematically speaking, a scheduling problem is then an optimization problem. In addition to an objective function, such a problem involves:

- **tasks**, representing the elementary units of the activity,
- **machines**, representing resources which are able to execute the tasks, (it can be real machines in a factory, processors in a computer as well as human operators),
- eventually other resources that tasks have to share, (like tool, energy, budget...),
- eventually resource constraints like capacity constraints,
- eventually additional time constraints, like precedence constraints between tasks, time windows during which tasks have to be processed, time windows during which machine are not available.

A solution of such a problem is called a **schedule**. A schedule assigns to each task a time period and a machine, and eventually other shared resources. A fundamental constraint that a schedule have to satisfy to be feasible, is the **task non-overlapping constraint**: two tasks cannot be processed on the same machine at the same time. Another one is the **non-negativity constraint** representing that we cannot plan to execute a task in the past, but only from now: a task cannot start before the time 0.

There exists a wide variety of scheduling problems, since they can model a lot of concrete problems appearing in agriculture, industry, services... The reader can refer to [10] and [33] for a wide range of examples. However, in this thesis, we will mainly focus on two single machine scheduling problems. Therefore, we present how single machine schedules are usually encoded and represented.

• *Encoding and representing a single machine schedule*

Let us consider a single machine framework, where a set of tasks J have to be processed non-preemptively. A schedule assigns to each task j an execution period, which is a time interval usually denoted by $[S_j, C_j]$. $\boldsymbol{S_j}$ (resp. $\boldsymbol{C_j}$) is called the **starting time** (resp. the **completion time**) of task j. Note that starting time and completion time are not time lengths but time points, that are dates. Assuming in addition that **processing times** are fixed, and denoted by $(\boldsymbol{p_j})_{j\in J}$, a schedule can be encoded by the vector of task completion times, *i.e.* $(C_j)_{j\in J}$, since for each task $j \in J$, we have $S_j = C_j - p_j$. Encoding schedules by the task completion times allows to express a wide range of constraints and objective functions.

Feasible schedules are commonly represented by machine oriented Gantt charts. In such a chart, each task $j \in J$ is represented by a rectangle, whose length represents its processing time, *i.e.* p_j, and each machine corresponds to an oriented axis representing the time horizon. Figure 1 gives the Gantt chart of $\mathcal{S}^1$ and $\mathcal{S}^2$, two illustrative schedules for the 4 tasks depicted at the top of the figure.

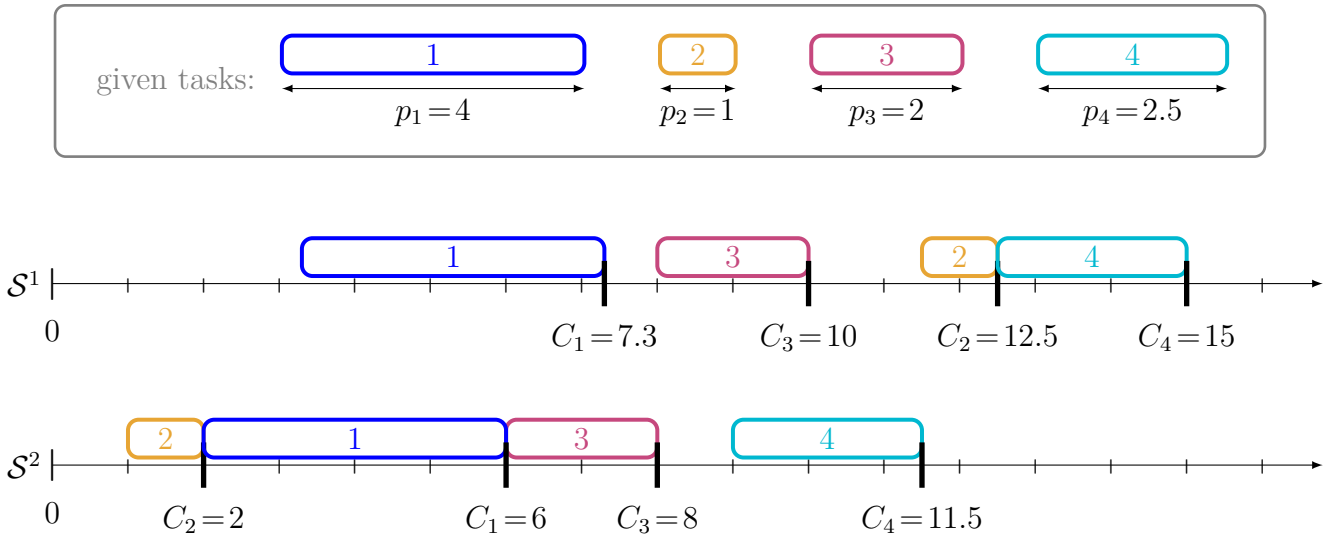


Figure 1: The Gantt charts of two illustrative 4-task schedules

• *Earliness-tardiness scheduling*

The just-in-time scheduling intends to reduce two kind of costs. On the one hand the storage costs, appearing when a product is completed before its delivery date, on the other hand the delay costs, appearing when a product is delivered after its delivery date. Because of the time needed for delivery, or because a task can be an intermediate operation in the manufacturing process of a final product,

we will not use delivery dates. We will rather use, for each task j, a **due date** denoted by $\boldsymbol{d_j}$, modeling the preferred completion time for j. For example, if a product has to be delivered at hour h for a client an hour's drive away, the due date of the last operation of this product is $h-1$.

If task j completes before its due date, then the earliness of j, is the duration between the completion time and the due date, *i.e.* $d_j - C_j$. Conversely, if j completes after d_j, then its earliness is 0. To cover these two cases, we can define the **earliness** as $\boldsymbol{E_j} = [d_j - C_j]^+$, where $[x]^+$ denotes the positive part of $x \in \mathbb{R}$. Similarly, the **tardiness** of task j, denoted by $\boldsymbol{T_j}$, is the length of time between the due date and the completion time if j completes after its due date, and 0 otherwise, *i.e.* $T_j = [C_j - d_j]^+$. Note that earliness and tardiness are not exactly symmetrical, as the processing time of a task is included in its tardiness but not in its earliness. Figure 2 illustrates on a Gantt chart the earliness of an early task and the tardiness of a tardy task sharing the same due date.

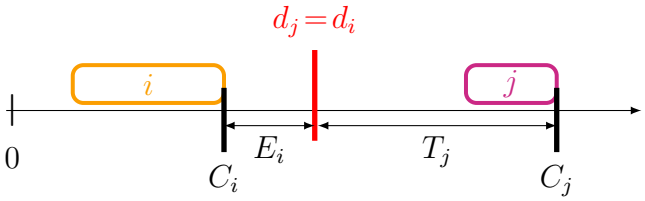


Figure 2: Earliness and tardiness on a Gantt chart

The reader can refer to [24] for a complete literature review on just-in-time scheduling. In this thesis, we will focus on problems where earliness and tardiness are linearly penalized. For each task $j \in J$, two coefficients are given: $\boldsymbol{\alpha_j}$ the unit earliness penalty, and $\boldsymbol{\beta_j}$ the unit tardiness penalty. Each time unit of earliness (resp. tardiness) of task j induces a penalty of α_j (resp. β_j). The total earliness-tardiness penalty of a whole schedule is then the following.

$$\sum_{j \in J} \alpha_j E_j + \beta_j T_j$$

In spite of the above linear writing, note that this objective function is not a linear function of the completion times. Indeed, the objective function can also be written as follows.

$$\sum_{j \in J} \alpha_j [d_j - C_j]^+ + \beta_j [C_j - d_j]^+$$

Figure 3 illustrates earliness (resp. tardiness) penalty for a given task $j \in J$ as a function of its completion time C_j. This function is piecewise linear, and its slope is defined by the unit earliness penalty α_j (resp. the unit tardiness penalty β_j).

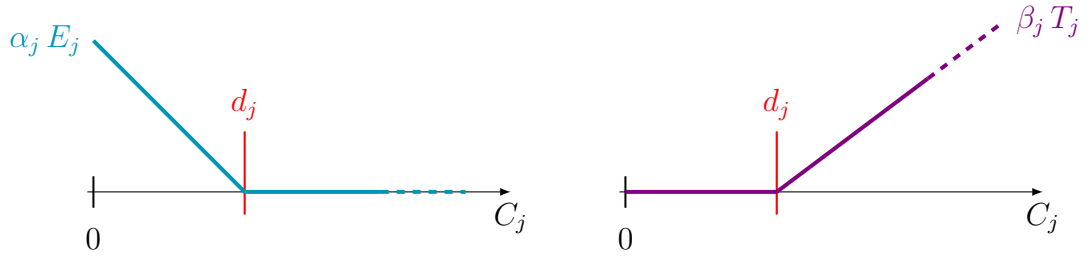


Figure 3: Earliness and tardiness penalties of a task j as a function of its completion time C_j

• *Common due date problems*

We will consider two **common due date** problems, that is where all the tasks share the same due date, which is then only denoted by $\boldsymbol{d}$, *i.e.* $\forall j \in J$, $d_j = d$. The objective function for the parameters $(\alpha, \beta) \in {\mathbb{R}_+}^2$ and $d \in \mathbb{R}_+$, denoted by $f_{\alpha,\beta,d}$ is then the following.

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

In practice, a common due date problem can reflect a situation where all the clients want to receive their products at the same date, for example when people order their cake to a pastry cook for Sunday 11:00 am. In a more industrial context, we can imagine the following situation: a joiner wants to send its daily production using an electric truck which leaves at 04:00 pm. Since some furniture is bulky, finishing a piece of furniture early will clutter up the small workshop. Earliness penalties reflect this discomfort. Conversely, if a piece of furniture is completed after 04:00 pm, the carpenter could pay an extra cost for a bike delivery: the longer the time to complete, the more the employee has to be payed for its overtime. Tardiness penalties reflect this cost. Finally, minimizing the total earliness-tardiness penalty consists in finding a day's work organization which is the best with regard to both the workshop congestion and extra delivery costs. These two objectives are aggregated in a linear way, then one can give more importance to the one or to the other by tuning the coefficients α_j and β_j.

Moreover, this problem can be seen as part of a problem with distinct due dates if a subset of tasks share the same due date.

The **general common due date problem** (CDDP) aims at finding a schedule, *i.e.* $C = (C_j)_{j \in J}$ minimizing the total earliness-tardiness penalty, *i.e.* $f_{\alpha,\beta,d}$, when all the tasks share the same due date.

CDDP

Input: a number of tasks $n \in \mathbb{N}$
the task processing times $(p_j)_{j \in [1..n]} \in \mathbb{R}_+^n$
the task unit earliness penalties $(\alpha_j)_{j \in [1..n]} \in \mathbb{R}_+^n$
the task unit tardiness penalties $(\beta_j)_{j \in [1..n]} \in \mathbb{R}_+^n$
a common due date $d \in \mathbb{R}_+$

Output: a feasible schedule $C = (C_j)_{j \in [1..n]}$ minimizing $f_{\alpha,\beta,d}(C)$

The instance of CDDP defined by its input parameters will be denoted by CDDP(p, α, β, d).

The **unrestrictive common due date problem** (UCDDP) is a special case of CDDP when the due date is **unrestrictive** , that is when $d \geqslant p(J)$, where for any subset $I \subseteq J$, and any vector $a \in \mathbb{R}^J$ indexed by J, $a(I)$ denotes the sum $\sum_{j \in I} a_j$. That means that the due date does not restrict the total duration of early tasks. This problem will be denoted by UCDDP, and a given instance of this problem by UCDDP(p, α, β, d).

As an illustration of CDDP and UCDDP instances we provide on page 13 the optimal schedule of 4 given tasks for different due dates.

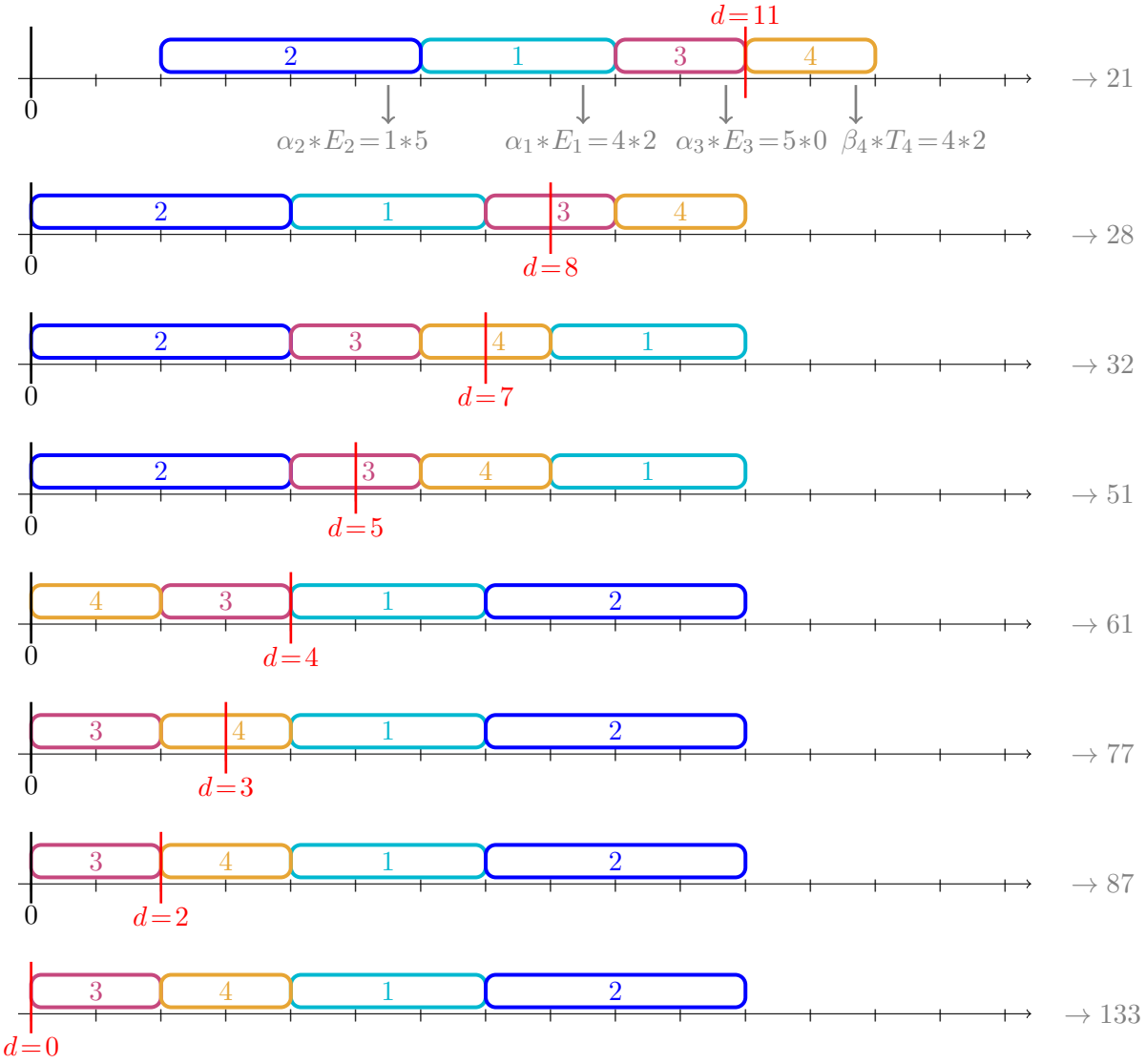
Different exact methods have been proposed to solve these problems (Cf. Section 0.3). Most of the proposed algorithms are based on dominance properties. Therefore, we present in the next section the dominance properties standing for UCDDP and CDDP, before presenting the resulting algorithms.

Example 1 : *Optimal schedules for different 4-task instances*

We consider the set of tasks $J=[1..4]$ defined by the parameters on the right. Depending on the value chosen for d, these parameters define an instance of UCDDP and CDDP, or an instance of CDDP only.
The following figure presents the[1]optimal schedules for different values of d.

$p_1=3,\ \alpha_1=4,\ \beta_1=5$
$p_2=4,\ \alpha_2=1,\ \beta_2=6$
$p_3=2,\ \alpha_3=5,\ \beta_3=8$
$p_4=2,\ \alpha_4=2,\ \beta_4=4$

For each schedule, the total earliness-tardiness penalty is indicated in gray on the right. For example it is 21 for the first schedule. Moreover, for the first schedule only, the penalty induced by each task is detailed in gray. For example, task 2 is early since it completes 5 time units before d, then $E_2=5$ and it induces an earliness penalty of $\alpha_2*E_2=1*5$.



Note that all the optimal schedules presented here have a common structure: the tasks are processed consecutively, without idle time. Moreover, in each schedule, there is a task starting at time 0, or a task completing at time d (or both). This is not a coincidence, there always exists an optimal schedule having this structure. Such properties, called dominance properties, are given in the next section.

In spite of their common structure, the optimal schedules can have completely different sequences, that are different task execution orders, when the value of d changes, even slightly. For example, task 2 is first executed in the optimal schedule for $d=5$, but last executed for $d=4$.

[1]For each instance given in this example, there is a unique optimal schedule, therefore we say "*the* optimal schedule". However, in general, there may be several optimal schedules.

0.2 Dominance properties for common due date problems

For a given optimization problem, and a given instance of this problem, 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 the optimal solutions. In both cases, the search of an optimal solution can be limited to the dominant set. For the sake of brevity, we say that a schedule is dominant if it belongs to a dominant set.

Other types of dominance properties exist: dominance between the instances of a given problem and dominance between problems for example. For an overview of the different kinds of dominance properties, illustrated with examples coming from the scheduling field, the reader can refer to [23]. Nevertheless, in this thesis, we use dominance property only to designate a property which states, for a given instance, that a solution subset is dominant or strictly dominant.

In order to describe dominant schedules, we first provide some definitions. For a given UCDDP or CDDP instance, we use α**-ratio** (resp. β**-ratio**) to designate $\alpha_j/p_j \in \overline{\mathbb{R}_+}$ (resp. $\beta_j/p_j \in \overline{\mathbb{R}_+}$). Moreover, for a given schedule $(C_j)_{j\in J}$, a task $j \in J$ is said **early** (resp. **on-time**, resp. **tardy**) if it completes at time $C_j \leqslant d$ (resp. $C_j = d$, resp. $C_j > d$). A task $j \in J$ is said **straddling** if it starts before d and completes after d, *i.e.* $C_j - p_j < d$ and $d < C_j$. Note that there is at most one on-time task[2] (resp. one straddling task), and that it is an early (resp. tardy) task. There cannot be both an on-time task and a straddling task. Finally, the **early-tardy partition** of the schedule is $(\boldsymbol{E}, \boldsymbol{T})$ where E (resp. T) is the early (resp. tardy) task subset. We then have $J = E \sqcup T$.

In just-in-time scheduling, an **idle time** designates a time period between two task execution periods during which no task is executed. We define a **block** as a feasible schedule without idle time, a d**-schedule** as a feasible schedule with an on-time task; a d**-block** as a d-schedule which is also a block, a **left-block** as a block starting at time 0, and a d**-or-left-block** as a block which is a d-schedule or which starts at time 0, or both. A schedule is said **V-shaped** [36] (resp. $\vee$**-shaped**) if early tasks are scheduled in non-decreasing order of their α-ratios and tardy tasks (resp. tasks starting after d) are scheduled in non-increasing order of their β-ratios. Note that being V-shaped or $\vee$-shaped is equivalent for schedules without straddling task.

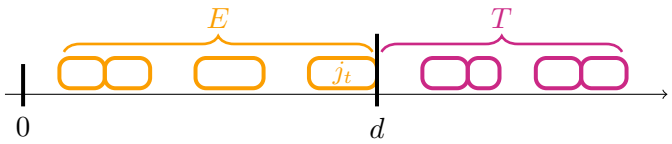


Figure 4: A d-schedule which is not a block

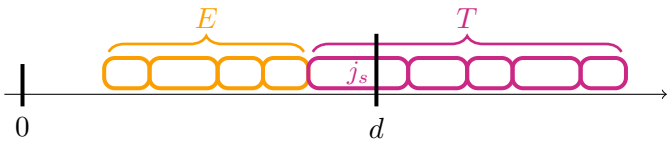


Figure 5: A block with a straddling task

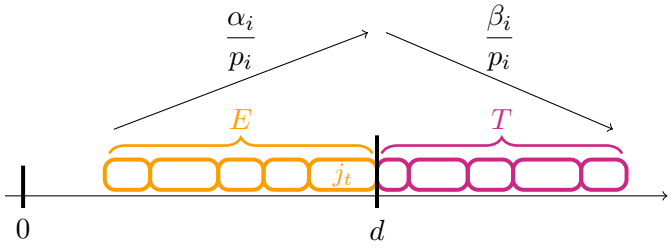


Figure 6: A V-shaped d-block

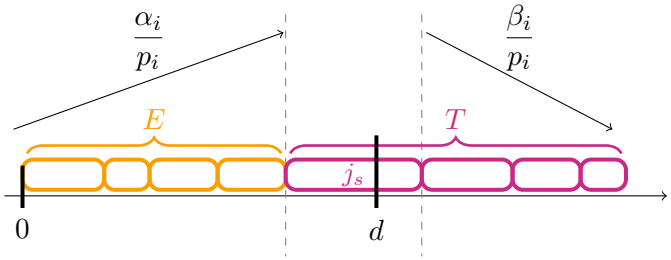
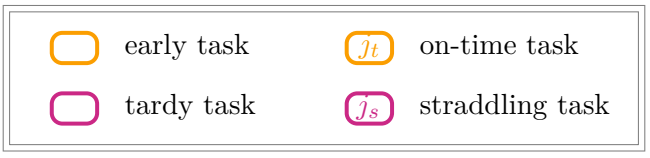


Figure 7: A $\vee$-shaped block starting at time 0



[2] except if there is a task with a zero processing time

0.2.1 Dominance properties for UCDDP

Some dominance properties for UCDDP with **symmetric penalties**, *i.e.* $\forall j \in J, \alpha_j = \beta_j$, are given in [21]. The following lemma extends these results to asymmetric penalties, using the same task shifting and exchange proof arguments.

Lemma 0.1

Let $p \in \mathbb{R}_+^J$ and $(\alpha, \beta) \in \mathbb{R}_+^J \times \mathbb{R}_+^J$. Let $d \in \mathbb{R}$ such that $d \geqslant p(J)$.

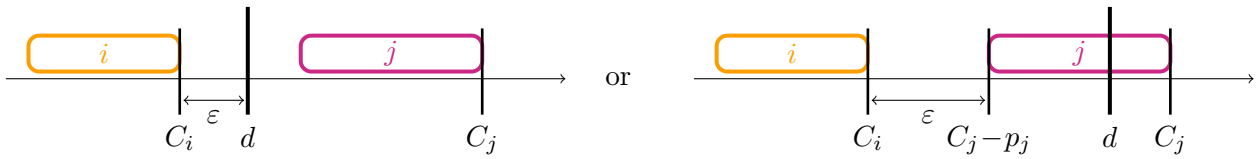
The set of d-blocks is dominant for UCDDP(p, α, β, d).

The set of blocks is even strictly dominant for UCDDP(p, α, β, d) if $(p, \alpha, \beta) \in {\mathbb{R}_+^*}^J \times {\mathbb{R}_+^*}^J \times {\mathbb{R}_+^*}^J$.

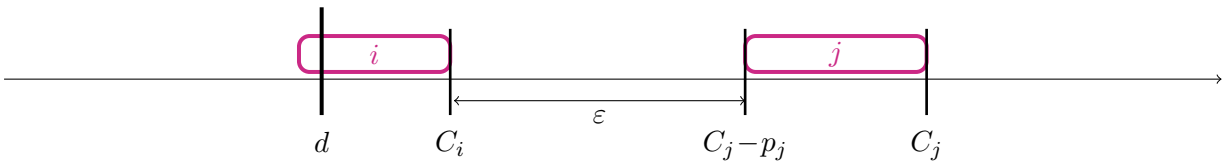
The set of V-shaped schedules is strictly dominant for UCDDP(p, α, β, d).

Proof of the block dominance property by shifting arguments : Let us assume that there exists an optimal schedule $\mathcal{S}^*$, encoded by the completion times $(C_j)_{j \in J}$, which is not a block, *i.e.* presenting an idle time. Necessarily there exists two tasks consecutive $(i, j) \in J^2$ in $\mathcal{S}^*$ such that there is an idle time between their execution periods, *i.e.* $C_j > C_i + p_j$ $(\star)$. There are two cases to consider according to the position of this idle time in relation to the due date.

- If $d > C_i$, we set $\varepsilon = \min\left(d - C_i, (C_j - p_j) - C_i\right)$. According to the assumption $(\star)$, we have $\varepsilon > 0$. We also introduce $I \subseteq J$ the subset of tasks placed before task i, including the task i itself. By right-shifting all the tasks in I by ε time units, these tasks stay early (since $C_i + \varepsilon \leqslant d$), but their earliness reduces by ε, inducing a total penalty reduction by $\varepsilon\,\alpha(I)$. If one task in I has a positive unit earliness penalty, we get a contradiction, since $\mathcal{S}^*$ is supposed to be optimal. Otherwise, $\alpha(I) = 0$ and this right-shifting results in another schedule having the same total penalty as $\mathcal{S}^*$, that is another optimal schedule.



- If $d \leqslant C_i$, we set $\varepsilon = (C_j - p_j) - C_i$. Once again, $\varepsilon > 0$ according to the assumption $(\star)$. We introduce $I \subseteq J$ the subset of tasks placed after task j, including the task j itself. By left-shifting all the task in I by ε time units, these tasks stay tardy (since $C_j - \varepsilon = C_i + p_j \geqslant d$), except if $p_j = 0$ and $C_i = d$, in which case task j becomes on-time. In all cases, the tardiness of each task in I reduces by ε, inducing a total penalty reduction by $\varepsilon\,\beta(I)$. If one task in I has a positive unit tardiness penalty, we get a contradiction, since $\mathcal{S}^*$ is supposed to be optimal. Otherwise, $\beta(I) = 0$ and this left-shifting results in another schedule having the same total penalty as $\mathcal{S}^*$, that is another optimal schedule.

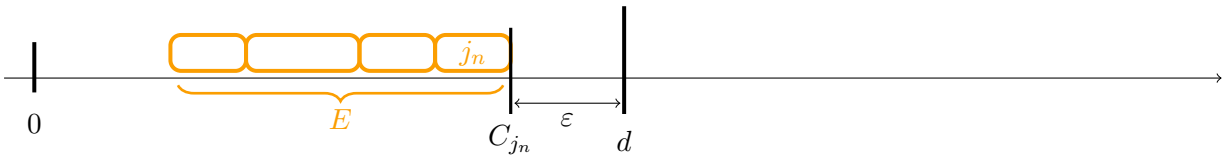


If $(p, \alpha, \beta) \in {\mathbb{R}_+^*}^J \times {\mathbb{R}_+^*}^J \times {\mathbb{R}_+^*}^J$, we get a contradiction in both cases. We deduce that there is no idle time in optimal schedules, hence the set of blocks contains all the optimal schedules.
If some instance coefficients are null, we cannot conclude to a contradiction, and for a good reason: there might exist optimal schedules with idle times. However, by iterating these shifting operations as long as idle times remain, we finally obtain an optimal block. That shows that blocks are dominant. This iterative procedure finishes since each idle time[3] is removed by one or two shifting operations. It is then necessarily an interval $[C_i, C_j - p_j]$ for two tasks i and j, and there are at most $n-1$ idle times in the initial schedule. □

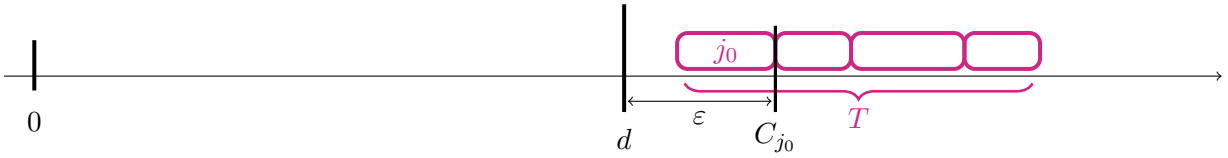
[3] In such a context, for a given schedule C, we call "an" idle time a maximal time interval between two task execution periods (maximal for $\subseteq$).

Proof of the d-block dominance property by shifting arguments : Here we want to show that an optimal schedule can always be found among the d-blocks. Thanks to the block dominance properties, there exists an optimal schedule $\mathcal{S}^*$, encoded by the completion times $(C_j)_{j\in J}$, which is a block. Let us assume that $\mathcal{S}^*$ is not a d-block. We denote by $j_0 \in J$ (resp. $j_n \in J$) the first (resp. last) task in $\mathcal{S}^*$. Hence, $C_{j_0}-p_{j_0}$ (resp. C_{j_n}) the starting time (resp. completion time) of the block. We also denote by (E,T) the early-tardy partition of this schedule. There are three cases to consider according to the position of d regarding to $C_{j_0}-p_{j_0}$ and C_{j_n}, knowing that $d\neq C_{j_n}$ since $\mathcal{S}^*$ is not a d-block.

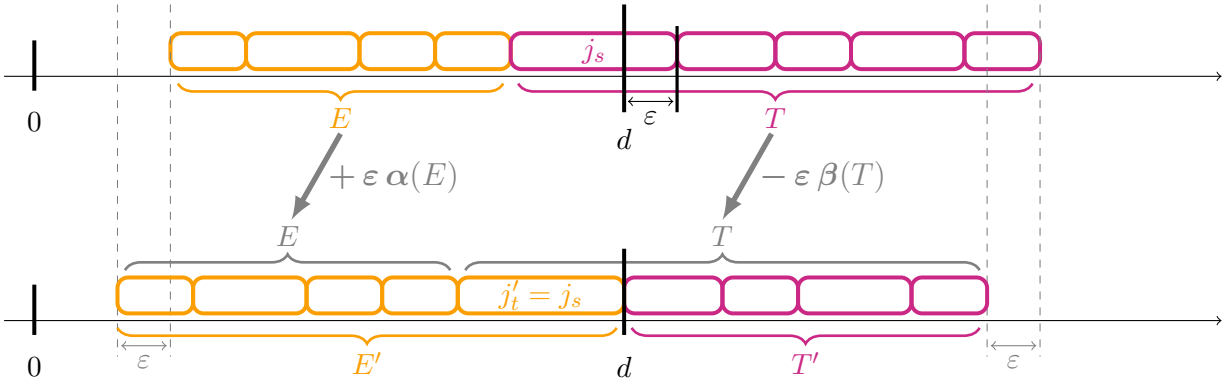
- If $d>C_{j_n}$, we set $\varepsilon=d-C_{j_n}$. We have $\varepsilon>0$. By right-shifting the whole block by ε time units all the tasks stay early, but their earliness reduces by ε inducing a total penalty reduction by $\varepsilon\,\alpha(J)$. Moreover, task j_n becomes on-time, then the obtained schedule is a d-block and is also an optimal schedule.



- If $d\leqslant C_{j_0}-p_{j_0}$, we set $\varepsilon=C_{j_0}-d$. We have $\varepsilon>0$ since $p_{j_0}>0$. By left-shifting the whole block by ε time units all the tasks stay tardy except task j_0 which becomes on-time. The tardiness of each task reduces of ε, inducing a total penalty reduction of $\varepsilon\,\beta(J)$. The obtained schedule, which is a d-block since j_0 is on-time, is then also an optimal schedule.



- If $d\in]C_{j_0}-p_{j_0}, C_{j_n}[$, there exists a straddling task $j_s\in J$, *i.e.* $C_{j_s}-p_{j_s}<d<C_{j_s}$, since the considered schedule is a block but not a d-block.
  - If $\alpha(E)<\beta(T)$, we set $\varepsilon=C_{j_s}-d$. By left-shifting the whole block by ε time units, the total tardiness penalty reduces by $\varepsilon\,\beta(T)$, while the total earliness penalty raises by $\varepsilon\,\alpha(E)$, which represents a reduction of the total penalty by $\varepsilon(\beta(T)-\alpha(E))>0$. Contradiction.

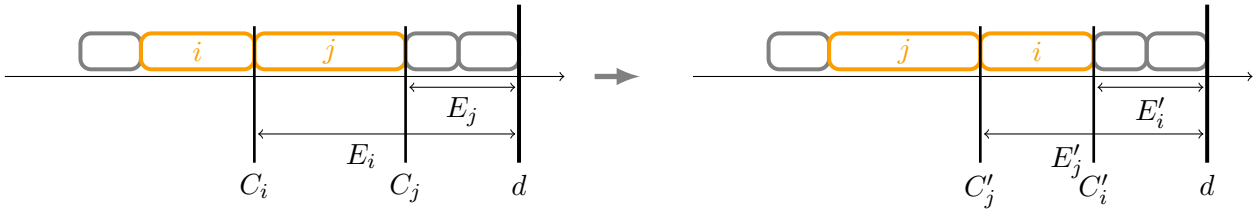


  - If $\alpha(E)>\beta(T)$, we set $\varepsilon=d-(C_{j_s}-p_{j_s})$. By right-shifting the whole block by ε time units, the total earliness penalty reduces by $\varepsilon\,\alpha(E)$, while the total tardiness penalty raises by $\varepsilon\,\beta(T)$, which represents a reduction of the total penalty by $\varepsilon(\beta(T)-\alpha(E))>0$. Contradiction.
  - If $\alpha(E)=\beta(T)$, there is an infinite number of optimal schedules which are not d-block: all those obtained by left-shifting the whole block by $\varepsilon<C_{j_s}-d$ and all those obtained by right-shifting the whole block by $\varepsilon<d-(C_{j_s}-p_{j_s})$. However, there also exists an optimal d-block, since there is at least the one obtained by a $C_{j_s}-d$ left-shifting in which j_s is on-time.

Finally, in the three cases an optimal d-block can be derived from the initial optimal block. □

Proof of the V-shaped dominance property by exchange arguments : Let us assume that there exists an optimal schedule $\widetilde{\mathcal{S}}$, which is not V-shaped. Using the block dominance property proof, we can transform $\widetilde{\mathcal{S}}$ into an optimal block $\mathcal{S}^*$ without changing neither the early-tardy partition, nor the task sequence. Let $(C_j)_{j\in J}$ denote the task completion times in $\mathcal{S}^*$. At least one of the two following cases occurs.

- Early tasks are not scheduled in the non-decreasing order of their α-ratio, then there exist two consecutive early tasks $(i,j)\in J$ such that i is placed before j, *i.e.* $C_i+p_j=C_j\leqslant d$, but has a larger α-ratio, *i.e.* $\frac{\alpha_i}{p_i}>\frac{\alpha_j}{p_j}$. By swapping tasks i and j, that is by completing task i at time C_j and task j at time $(C_i-p_i)+p_j$, the earliness of task i reduces by p_j time units, while the earliness of task j raises by p_i time units. That results in a reduction of the total penalty of $-\alpha_i\,p_j+\alpha_j\,p_i>0$. Contradiction.



- Tardy tasks are not scheduled in the non-increasing order of their β-ratio, then there exist two consecutive early tasks $(i,j)\in J$ such that i is placed before j, *i.e.* $d\leqslant C_i$ and $C_i+p_j=C_j$, but has a smaller β-ratio, *i.e.* $\frac{\beta_i}{p_i}<\frac{\beta_j}{p_j}$. By swapping tasks i and j, that is by scheduling task i to complete at time C_j and task j to complete at time $(C_i-p_i)+p_j$, the tardiness of task i raises by p_j time units, while the tardiness of task j reduces by p_i time units. That results in a reduction of the total penalty of $\beta_i\,p_j-\beta_j\,p_i>0$. Contradiction.

We deduce that $\mathcal{S}^*$ is V-shaped, then $\widetilde{\mathcal{S}}$ is V-shaped too. □

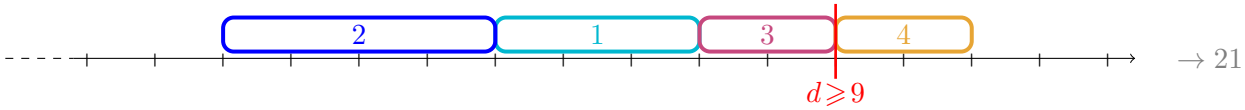
The Lemma 0.1's dominance properties are of major importance as all the results given in Section 0.3.2, as well as the formulations of UCDDP provided in Section 0.5 are derived from these dominance properties Moreover, using Lemma 0.1, we can make some assumptions on parameters p,α,β of the UCDDP without loss of generality.

- *Assuming that $d=p(J)$ without loss of generality*

An important point to notice is that the total earliness-tardiness penalty of a d-block only depends on the task sequence for both sides of the due date, no matter neither the value of the due date nor the task completion times. This allows to solve all the unrestrictive instances by solving only the one with $d=p(J)$ for instance. We illustrate this remark on an example, before using the dominance properties to prove it.

Example 1 (continued) : *Equivalence of all the unrestrictive instances*

Let us consider again the 4 tasks and the optimal schedules introduced in Example 1 on Page 13. In the optimal d-block for the unrestrictive instance with $d=p(J)=11$, the total length of early tasks is $\Delta=p_2+p_1+p_3=9$. Therefore, for any $d\geqslant\Delta$, shifting the whole block such that task 3 completes at d, as represented below, always provides a d-block of total penalty 21. In addition, this schedule is necessary optimal, which can be proved by a converse shifting argument.



Thanks to the d-block dominance, the optimal value does not depend on the value of d (provided that d is unrestrictive). Indeed, let us consider an instance UCDDP(p,α,β,d) where $d>p(J)$, and consider $d'=p(J)$ as a new due date. In addition, let us consider a d-block $\mathcal{S}$, and denote by (E,T) its early-tardy partition. By definition, $\mathcal{S}$ starts at time $s=d-p(E)$. Since $E\subseteq J$, we have $s-(d-d')=-p(E)+p(J)\geqslant 0$. That ensures that we can left-shift $\mathcal{S}$ by $d-d'$ time units without violating the non-negativity constraint. The obtained schedule $\mathcal{S}'$ is a d'-block, and each task has exactly the same earliness (resp. tardiness) in $\mathcal{S}$ with respect to d as in $\mathcal{S}'$ with respect to d'. It follows that $\mathcal{S}$ as a solution of UCDDP(p,α,β,d) and $\mathcal{S}'$ as a solution of UCDDP (p,α,β,d') have the same value. More generally, there exists a one-to-one correspondence preserving the objective function between the dominant sets for these two instances: solving UCDDP (p,α,β,d') is thus equivalent to solving UCDDP(p,α,β,d).

• *Assuming that* $\forall j\in J, p_j>0$ *without loss of generality*

Thanks to the d-block dominance, we can only consider tasks having a positive processing time. Indeed, let us assume that there exists a task $j\in J$ having a zero processing time, *i.e.* $p_j=0$. In any d-block, the task j can be moved at d, since no task is processing at time d (there is no straddling task). This move does not impact other tasks. Moreover, it does not increase the penalty induced by task j, since j incurs no penalty when it is placed at d. We deduce that the set of d-blocks where j is processed at d is dominant.
Since removing task j does not change the schedule penalty when j is placed at time d, there is a one-to-one correspondence preserving the total penalty between this dominant set and the set of the d-blocks for $J\backslash\{j\}$. We deduce that an optimal d-block for $J\backslash\{j\}$ corresponds to an optimal d-block for J. In other words, it suffices to solve the problem for the instance where the zero processing time task j has been removed.

• *Assuming that* $\forall j\in J, \beta_j\in\mathbb{R}_+^*$ *without loss of generality*

Moreover, we can only consider tasks having positive unit tardiness penalties. Indeed, let us assume that there is a task $j\in J$ having a zero unit tardiness penalty, *i.e.* $\beta_j=0$. In any d-block, the task j can be moved at the end of the schedule without impacting other tasks. This move does not raise the penalty induced by j, since placed tardy, j incurs no penalty. If this move produces an idle time, we transform the obtained schedule into a d-block by right-shifting some early tasks or by left-shifting some tardy tasks. In both cases, that does not raise again the penalty. We deduce that the set of d-blocks where j is processed last is dominant.
Since removing task j from such a schedule does not change its penalty, there is a one-to-one correspondence preserving the penalty between this dominant set and the set of the d-blocks for $J\backslash\{j\}$. We deduce that an optimal d-block for $J\backslash\{j\}$ corresponds to an optimal d-block for J. In other words, it suffices to solve the problem for the instance where the zero unit tardiness penalty task j has been removed.

• *Assuming that* $\forall j\in J, \alpha_j\in\mathbb{R}_+^*$ *without loss of generality*

Thanks to the unrestrictiveness of d, we can also only consider tasks having positive unit earliness penalties. We almost follow the same line as above, but we have here to use the assumption $d\geqslant p(J)$. Let us assume there is a task $j\in J$ with a zero unit earliness penalty, *i.e.* $\alpha_j=0$. Let us consider a d-block, and denote by (E,T) its early-tardy partition. We check that task j can be moved at the beginning of this d-block to obtain a d-block with a non-larger penalty.

- If $j\in T$, we want to place j such that j completes at $d-p(E)$ (and then necessarily starts at $d-p(E)-p_j$). Since $j\notin E$, we have $p(E)+p_j\leqslant p(J)$. Since $d\geqslant p(J)$, we deduce that $d-p(E)-p_j\geqslant 0$, that is processing a task between $d-p(E)-p_j$ and $d-p(E)$ respects both non-negativity and non-overlapping constraints. The task j can thus be moved at the beginning

of the d-block. This move does not increase the total penalty since, placed early, j incurs no penalty. By left-shifting some tardy tasks, we transform the obtained schedule into a d-block, without increasing the total penalty.

- If conversely $j \in E$, then we can first remove task j, then right-shift the early tasks placed before j so that there is no more idle time, which does not raise their earliness penalties, and finally place j at the beginning of the schedule, which induces no penalty since j is placed early. This operation produces a d-block with a lower penalty.

We deduce that the set of d-blocks where j is scheduled first is dominant. As previously done, we establish a one-to-one correspondence preserving the total penalty between this dominant set and the set of the d-blocks for $J \setminus \{j\}$, and deduce that it suffices to solve the problem for the instance where the zero unit earliness penalty task j has been removed.

Thanks to the previous remarks, an instance of UCDDP will be defined by $p \in {\mathbb{R}_+^*}^J$, $\alpha \in {\mathbb{R}_+^*}^J$ and $\beta \in {\mathbb{R}_+^*}^J$ in the sequel. The due date will be implicitly fixed at $d = p(J)$. Such an instance will be denoted UCDDP(p, α, β).

0.2.2 Dominance properties for CDDP

In the general case, the dominance of the d-blocks is no longer valid. Indeed, since the total length of the early tasks is limited by d, some tasks are tardy even if they would have a lower penalty scheduled early. Figure 8 shows that the penalty of the left-block[4] is smaller than the penalty of the d-block for two identical tasks characterized by $p_j=3$, $\alpha_j=2$, $\beta_j=4$, for $j\in\{1,2\}$

Figure 8: The two possible d-or-left-blocks for a 2-identical-task instance

The following lemma gives dominance properties already known for the common due date problem with symmetric penalties (*Cf.* [22] and [20]). These results have been extended to asymmetric penalties in [13], using the same task shifting and exchange proof arguments. The proof is omitted since it is very similar to Lemma 0.1's proof.

Lemma 0.2

Let $p\in\mathbb{R}^J$, $(\alpha,\beta)\in\mathbb{R}_+^J\times\mathbb{R}_+^J$, and $d\in\mathbb{R}_+$.
The set of d-or-left-blocks is dominant for CDDP(p,α,β,d).
The set of $\vee$-shaped schedules is strictly dominant for CDDP(p,α,β,d).

Note that the $\vee$-shaped property gives no information about the straddling task β-ratio. This remark is more fully discussed in Section 0.5.3.

- *What we can assume about CDDP instance parameters without loss of generality*

In a CDDP instance, the due date can be restrictive (*i.e.* $d<p(J)$). In this case left-shifting a d-block by $d-p(J)$ time units can result in a schedule starting before time 0, which is thus unfeasible. Therefore, we cannot assume that $d=p(J)$ for CDDP instances. Similarly, the arguments used to show that we can assume that $\forall j\in J,\alpha_j>0$ are not still valid for a restrictive due date. We provide a counter-example below (*Cf.* page 21)

Conversely, the arguments establishing that we can assume that $\forall j\in J,p_j>0$ and $\beta_j>0$ are still valid for an unrestrictive due date.

> Thanks to the previous remarks, in the sequel a CDDP instance will be defined by $p\in{\mathbb{R}_+^*}^J$, $\alpha\in{\mathbb{R}_+}^J$, $\beta\in{\mathbb{R}_+^*}^J$ and $d\in\mathbb{R}_+$.

[4]Since the two tasks are identical, the two possible left-blocks are identical, idem for the two possible d-blocks.

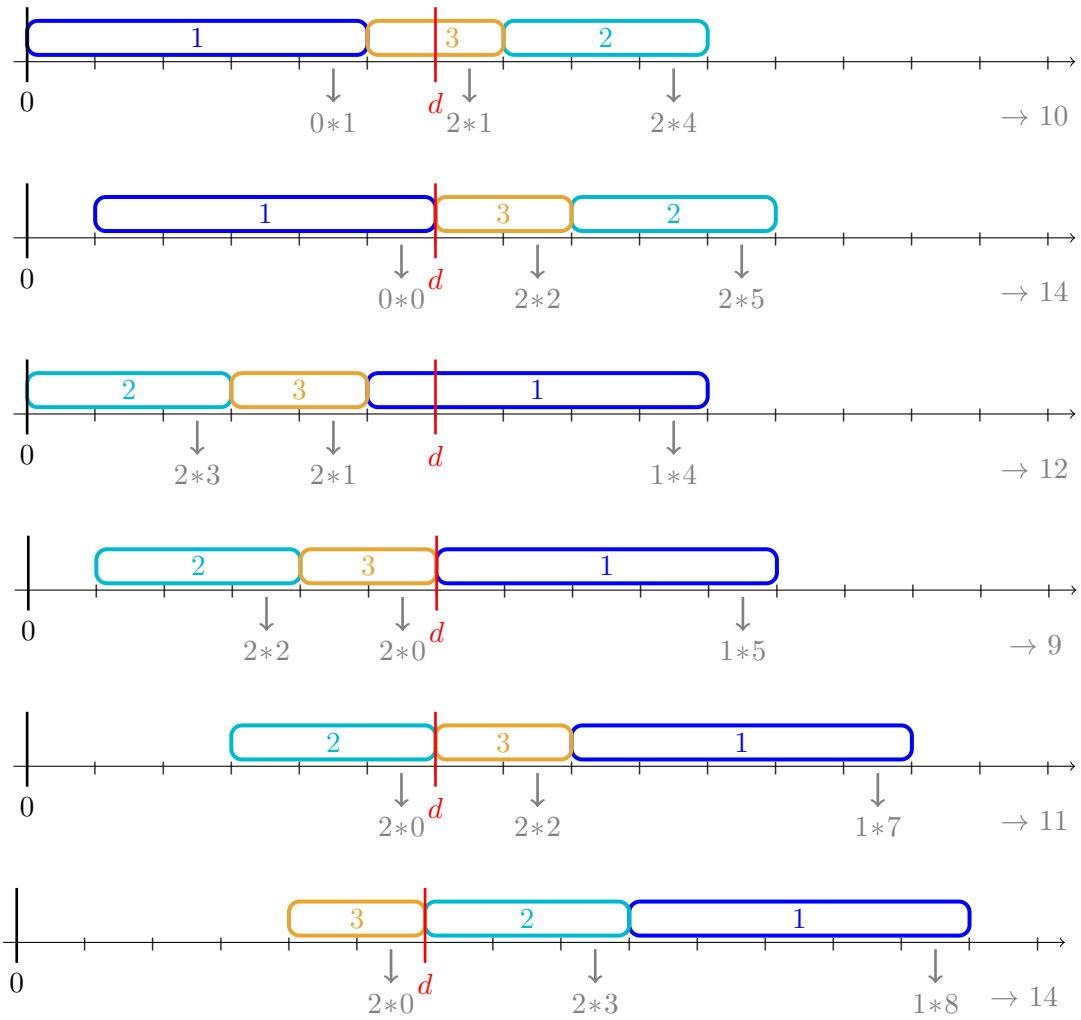
Counter-example 1 : *We cannot assume without loss of generality that* $\forall j \in J, \alpha_j > 0$ *for* CDDP

Let us consider the following instance of CDDP with a task having a 0 unit earliness penalty (task 1).

$$J=[1..3],\ d=6,\ p_1=5,\ \alpha_1=0,\ \beta_1=1,$$
$$p_2=3,\ \alpha_2=2,\ \beta_2=2,$$
$$p_3=2,\ \alpha_3=2,\ \beta_3=2,$$

For this instance it is not possible to follow the solving approach described on page 18 by setting aside task 1. Indeed, if we solve the problem for $J=\{2,3\}$, we may obtain the optimal solution where task 2 is on-time, and task 3 starts at time d (another optimal solution starting earlier exists). This schedule starts at time 3, which does not allow to place task 1 of length 6 before while satisfying the non-negativity constraint.

According to Lemma 0.2, an optimal schedule can be found among the $\vee$-shaped d-or-left-blocks. The following figure presents the $\vee$-shaped d-or-left-blocks for this instance. For each schedule, the total penalty is written in gray on the right, moreover the task penalties are detailed under each one. One can check that all the $\vee$-shaped d-or-left-blocks are represented. Indeed, in such a schedule, if they are early, task 1 must be placed before task 2, itself placed before 3 since $\frac{\alpha_1}{p_1}=0<\frac{\alpha_2}{p_2}=\frac{2}{3}<\frac{\alpha_3}{p_3}=1$, while if they are tardy, task 3 must be placed before task 2, itself placed before 1 since $\frac{\beta_3}{p_3}=1>\frac{\beta_2}{p_2}=\frac{2}{3}>\frac{\beta_1}{p_1}=\frac{1}{5}$.



The optimal schedule with a total penalty equals to 9, is a d-block where task 1 is tardy. Moreover, the schedules where task 1 is early are not optimal. That shows that we cannot say that it is always better to schedule early the tasks having a zero unit earliness penalty.

0.3 Algorithms for the common due date problems

This section presents different algorithms provided in the literature for solving some common due date problems, along with complexity results. Several sub-problems of UCDDP and CDDP, obtained by making assumptions on α and β, are considered. Other related problems are presented.

0.3.1 A polynomial algorithm for UCDDP when unit earliness and tardiness penalties do not depend on the task

• *Kanet algorithm*

Kanet [26] considers UCDDP in the particular case where all the unit earliness or tardiness penalties are equal, *i.e.* $\forall j \in J$, $\alpha_j = \beta_j = c$ for $c \in \mathbb{R}$. This minimization problem, which does not depend on the constant c, is also called the *mean absolute deviation minimization problem*, since for $c = \frac{1}{n}$ the objective function can be written as follows.

$$\sum_{j\in J} \alpha_j E_j + \beta_j T_j \;=\; \frac{1}{n}\sum_{j\in J} E_j + T_j \;=\; \frac{1}{|J|}\sum_{j\in J} [d - C_j]^+ + [C_j - d]^+ \;=\; \frac{1}{|J|}\sum_{j\in J} |C_j - d|$$

However, for the sequel, we assume that $c = 1$. Using V-shaped d-block dominance property, Kanet [26] provides for this problem an exact $\mathcal{O}(n\log(n))$ algorithm Let us present this algorithm. Once the number of early (resp. tardy) tasks n_E (resp. $n_T = n - n_E$) is fixed, we know that, for $k \in [1..n_E]$ (resp. $k \in [1..n_T]$), placing j at the k-th (resp. $n+1-k$-th) position in a d-block contributes for $(k–1)\, p_j$ (resp. $k\, p_j$) to its total penalty. Finding an optimal schedule for (n_E, n_T) is then equivalent to an assignment problem: it then suffices to sort tasks by decreasing processing times and assign them to the lower penalty positions, that is going from the external positions to the central ones by alternating the right and left side of d. Following this assignment procedure, the obtained schedule is a V-shaped d-block, whose total penalty depends on (n_E, n_T). Using the expression of this total penalty, one can ensure that it is minimal for $n_E = \lceil \frac{n}{2} \rceil$ and $n_T = \lfloor \frac{n}{2} \rfloor$. On page 23, we give a proof of these results within a more general framework where earliness and tardiness can have different weights. The following example illustrates the Kanet's algorithm course.

Example 3 : *The Kanet's algorithm course on a 4-task instance*

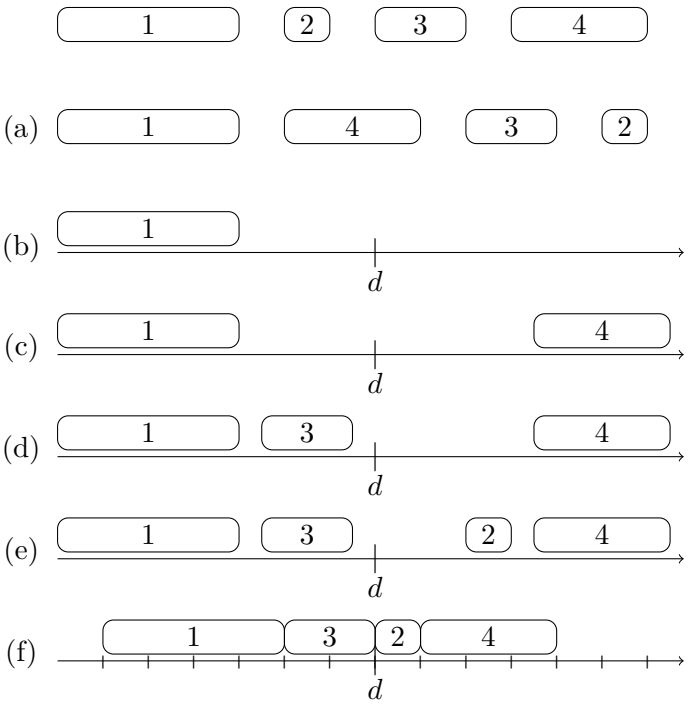
We consider the 4-task instance depicted on the top right, which is formally defined by $J = \{1, 2, 3, 4\}$, $p_1 = 4$, $p_2 = 1$, $p_3 = 2$, $p_4 = 3$.

The first step of the algorithm is to sort the tasks by decreasing processing times (a).

Then the sequence is fixed by distributing tasks, from the external position to the central position, alternating early and tardy side, starting on the early one (b – e).

Here, the longest task 1 is placed on the early side, at the first position (b). Then the second longest task 4 is placed on the tardy side, at the last position (c). The third longest task 3 is placed on the early side, between the already placed early tasks and d, that is at the second position (d). The shortest task 2, is placed on the tardy side, between d and the already placed tardy tasks, that is in second last position (e).

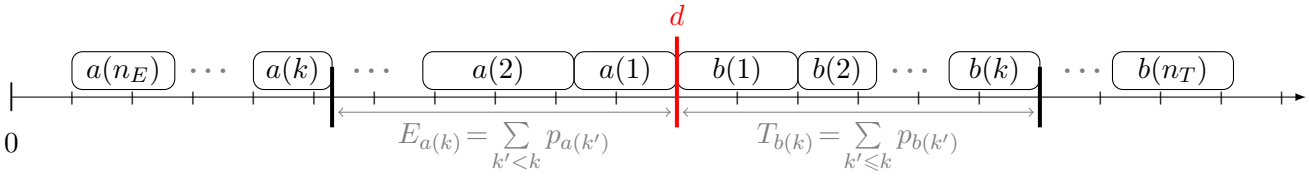
The schedule is finally obtained from this sequence by gathering the tasks around d (f).



- *Extension of Kanet's algorithm for distinct earliness and tardiness penalties*

Bagchi, Chang, and Sullivan [4] adapt the Kanet's algorithm to handle instances where unit earliness and tardiness penalties are distinct but still do not depend on the tasks, *i.e.* $\forall j \in J, \alpha_j = \alpha_0 \in \mathbb{R}_+^*$ and $\forall j \in J, \beta_j = \beta_0 \in \mathbb{R}_+^*$. As Kanet did for the case where $\alpha_0 = \beta_0$, they provide an $\mathcal{O}(n \log(n))$ algorithm which consists in first sorting the tasks by non-decreasing processing times, and then place them successively, in E if $\alpha_0 |E| < \beta_0 (|T|+1)$ and in T otherwise. This algorithm results in a d-block where the number of tardy tasks n_E is γ or $\gamma+1$ if $\boldsymbol{\gamma} = n \frac{\beta_0}{\alpha_0+\beta_0}$ is integer, and is $\lceil \gamma \rceil$ otherwise. We provide below a proof of this result. Note that this proof is different from the one proposed in [4].

Proof: Let $(n_E, n_T) \in \mathbb{N}^2$ such that $n_E + n_T = n$. Let $\mathcal{S}$ a schedule having exactly n_E early tasks and n_T tardy ones. Let $a \in \mathcal{F}([1..n_E], J)$ (resp. $b \in \mathcal{F}([1..n_T], J)$) such that $a(k)$ (resp. $b(k)$) is the task placed at the k-th position before d (resp. after d) in $\mathcal{S}$.



The total penalty of $\mathcal{S}$ can then be written as follows.

$$\begin{aligned}
\alpha_0 \sum_{j \in J} E_j + \beta_0 \sum_{j \in J} T_j = \alpha_0 \sum_{k=1}^{n_E} E_{a(k)} + \beta_0 \sum_{k=1}^{n_T} T_b(k) \;&=\; \alpha_0 \sum_{k=1}^{n_E} \sum_{k'=1}^{k-1} p_{a(k')} + \beta_0 \sum_{k=1}^{n_T} \sum_{k'=1}^{k} p_{b(k')} \\
&= \alpha_0 \sum_{k'=1}^{n_E} \sum_{k=k'+1}^{n_E} p_{a(k')} + \beta_0 \sum_{k'=1}^{n_T} \sum_{k=k'}^{n_T} p_{b(k')} \\
&= \alpha_0 \sum_{k'=1}^{n_E} (n_E - k')\, p_{a(k')} + \beta_0 \sum_{k'=1}^{n_T} (n_T - (k'-1))\, p_{b(k')}
\end{aligned}$$

The problem of finding a schedule $\mathcal{S}$ with n_E early tasks and n_T tardy ones that minimizes the total penalty is thus equivalent to find how to assign each p_j, either to an early position $k \in [1..n_E]$ whose cost is $\alpha_0 (n_E - k)$, or to a tardy position $k \in [1..n_T]$ whose cost is $\beta_0 (n_T - k + 1)$. For given (n_E, n_T), let us denote by $z(n_E, n_T)$ the cost of optimal assignments. The value of an optimal schedule (for the initial problem) is then $\min\{z(n_E, n_T) \,|\, n_E + n_T = n\}$. Using exchange arguments, one can prove that an optimal solution for such assignment problem is obtained by assigning the longest processing times to the lowest costs. Moreover, one can note that the more (n_E, n_T) offers positions with a small cost, the smaller is the cost of the best assignment, *i.e.* $z(n_E, n_T)$. In particular, this argument allows us to deduce the following necessary conditions on $(n_E, n_T) \in \mathbb{N}^2$ to minimize z.

$$\begin{aligned}
(n_E, n_T) \in \underset{u+v=n}{\arg\min}\, z(u,v) &\Rightarrow \begin{cases} z(n_E, n_T) \leqslant z(n_E+1, n_T-1) \\ z(n_E, n_T) \leqslant z(n_E-1, n_T+1) \end{cases} \\
&\Rightarrow \begin{cases} \alpha_0\, n_E \geqslant \beta_0\, n_T \\ \beta_0\, (n_T+1) \geqslant \alpha_0\, (n_E-1) \end{cases} \\
&\Rightarrow \begin{cases} \alpha_0\, n_E \geqslant \beta_0\, (n-n_E) \\ \beta_0\, ((n-n_E)+1) \geqslant \alpha_0\, (n_E-1) \end{cases} \\
&\Rightarrow \begin{cases} (\alpha_0+\beta_0)\, n_E \geqslant \beta_0\, n \\ \beta_0\, n \geqslant (\alpha_0+\beta_0)\, (n_E-1) \end{cases} \\
&\Rightarrow \begin{cases} n_E \geqslant \gamma \\ \gamma \geqslant (n_E-1) \end{cases} \\
&\Rightarrow n_E \in [\gamma, \gamma+1] \cap \mathbb{N}
\end{aligned}$$

Note that z necessary admits a minimizer over the finite set of possible couples (n_E, n_T). Moreover, one can check that the value of z is the same for all couples $(n_E, n-n_E)$ such that $n_E \in [\gamma, \gamma+1] \cap \mathbb{N}$. Therefore, the previous implications are in fact equivalences.

This proves that the original scheduling problem can be solved by first fixing (n_E, n_T) as presented according to the parameters α_0, β_0 and n, and then assigning the different processing times to the different positions for the costs mentioned above. □

• *What is really an unrestrictive instance for Kanet algorithm*

One can note that the sequence of the d-block provided by Kanet's algorithm does not depend on the due date $d \geqslant p(J)$. More precisely, neither the sequence, nor the on-time task depend on d. Indeed, for a larger due date $d' \geqslant d$, it suffices to right-shift the whole d-block by $d'-d$ time units to obtain an optimal schedule. For a smaller due date $d' \leqslant d$, we can similarly left-shift the whole d-block by $d-d'$ time units, assuming that d' is larger than the total length of early tasks, otherwise this left-shifting produces a schedule which does not satisfy the non-negativity constraint.

Let us define a **sequence** as a total strict order on the set of tasks, a **pointed sequence** as a sequence provided with an on-time task. One can consider that Kanet's algorithms takes as only input the processing times, and that it provides as output a **pointed sequence**, rather than a schedule. The optimal schedule is directly deduced from this pointed sequence as soon as a large enough due date is given.

A priori the minimal suitable value for the due date is simply the total length of the early[5] tasks in the pointed sequence. However, the sequence can be chosen so as to minimize this length, as proposed by Bagchi, Chang, and Sullivan in [5]. Indeed, for each $k \in [1..\lfloor \frac{n}{2} \rfloor]$, the $(k+1)$-th position and the $(n+1-k)$-th position in the sequence have the same cost, namely k. Therefore, when the only goal is to find a minimum penalty sequence, like in the Kanet's algorithm, the $2k$-th and the $(2k+1)$-th longest tasks can be placed arbitrarily at these two positions, since it results in two pointed sequences having the same penalty. On the contrary, Bagchi, Chang, and Sullivan impose how to place these two tasks in order to minimize the total length of early tasks: the shortest (*i.e.* the $2k$-th longest tasks of the instance) has to be placed early (*i.e.* at the $(k+1)$-th position) and the longest (*i.e.* the $(2k+1)$-th longest tasks of the instance) has to be placed tardy (*i.e.* at the $(n+1-k)$-th position).

Let us denoted by Δ the total length of early tasks in the resulting sequence. For any $d \geqslant \Delta$, this $\mathcal{O}(n \log n)$ algorithm provides an optimal solution, therefore one can say that an instance is *"really restrictive"* when $d < \Delta$. However, in the sequel, we keep the definition of unrestrictive due date given previously, that is $d \geqslant p(J)$.

[5]By extension of the definition for a schedule, the early tasks of a pointed sequence are those place before the on-time task in the sequence, including the on-time task itself, and the tardy tasks are the other tasks.

0.3.2 A pseudo-polynomial algorithm for UCDDP when unit earliness and tardiness penalties are symetric

Hall and Posner [21] study UCDDP in the case of symmetric earliness-tardiness penalties, *i.e.* $\forall j \in J,\ \alpha_j = \beta_j$. In this case, the unit penalties are simply denoted $(w_j)_{j\in J}$. Moreover, they assume that processing times are integer. The problem is then called **weighted earliness-tardiness problem** (WETP).

WETP

Input: a number of tasks $n \in \mathbb{N}$
the task processing times $(p_j)_{j\in[1..n]} \in \mathbb{N}^n$
the task unit earliness-tardiness penalties $(w_j)_{j\in[1..n]} \in \mathbb{R}_+^n$
a common due date $d \in \mathbb{R}_+$ such that $d \geqslant p(J)$

Output: a feasible schedule $C = (C_j)_{j\in[1..n]}$ minimizing $f_{w,w,d}(C)$

- *Complexity results*

Hall and Posner [21] show that WETP is NP-hard by reduction from the **even-odd partition problem** (EOPP), which is shown be NP-hard in [16].

EOPP

Input: $(a_i)_{i\in[1..2n]} \in (\mathbb{N}^*)^{2n}$ sorted in increasing order

Output: yes if there exists a bi-partition $\{U, V\}$ of $[1..2n]$ s.t $\begin{cases} \sum_{i\in U} a_u = \sum_{v\in V} a_v \\ \forall k \in [1..n], \{2k\text{-}1, 2k\} \cap U \neq \emptyset \\ \forall k \in [1..n], \{2k\text{-}1, 2k\} \cap V \neq \emptyset \end{cases}$
no otherwise

Since UCDDP is more general than WETP, that implies that UCDDP is NP-hard. More precisely UCDDP is weakly NP-hard since it can be solved by a pseudo-polynomial algorithm proposed in [21] (*Cf.* below).

- *A pseudo-polynomial algorithm based on dynamic programming*

Hall and Posner [21] propose a dynamic programming algorithm to exactly solve UCDDP for symmetric penalties. Let us present this algorithm based on the V-shaped d-block dominance property. Let us assume that tasks are sorted by non-increasing w-ratio, *i.e.* w_j/p_j does not increase when j raises. Without loss of generality, we consider only the d-blocks for which early (resp. tardy) tasks are scheduled in decreasing (resp. increasing) order of their indices. Note that, in such schedules, for any $k \in [1..n+1]$, tasks $[1..k-1]$ are placed consecutively around d: in other words, they form a d-block.

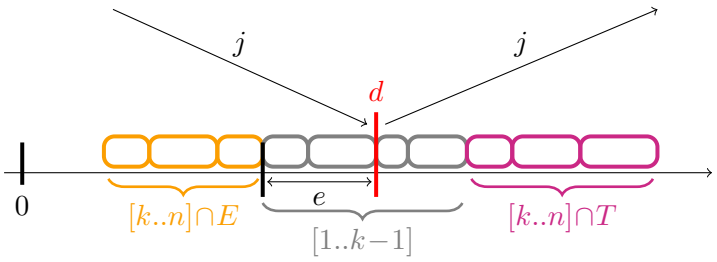


Figure 9: A dominant schedule for the Hall and Posner's algorithm

Moreover, the penalty induced by tasks $[k..n]$ does not depends of the task sequence of this d-block, but only of its starting and completion times, which can be deduced from each other. If $e = p\big(E \cap [1..k-1]\big)$, then the starting time of the d-block is $d-e$ while its completion time is $d-e+p\big([1..k-1]\big)$ (which is equal to $d\ + p\big(T \cap [1..k-1]\big)$. Note that $e \in [0..p(J)]$ since processing

times are integers.

Thanks to these observations, for any $k \in [1..n+1]$, the minimal penalty induced by tasks $[k..n]$ in a dominant schedule only depends on e. It is then denoted by $\boldsymbol{f_k(e)}$. By definition, $f_1(0)$ is the total penalty of an optimal schedule. Moreover, for any $e \in [0..p(J)]$, $f_{n+1}(e) = 0$ since $[n+1..n] = \emptyset$. The value of $f_k(e)$ for $e \in [0..p(J)]$ and $k \in [1..n]$ can be computed by reverse induction using the following recurrence relation.

$$f_k(e) = \min\Big(f_{k+1}(e+p_k) + w_k\, e,\ f_{k+1}(e) + w_k\ \Big(p\Big([1..k]\Big) - e\Big)\Big)$$

Indeed, let us assume that tasks $[1..k-1]$ are already scheduled so as to form a d-block starting e time units before d, and consider a schedule $\mathcal{S}$ optimally completing this block. According to previous observations, task k can only hold at two positions.

→ Either k is placed just before the block, that is with an earliness of e. In this case, k induces an earliness penalty of $w_k\, e$, and, by optimality of $\mathcal{S}$, tasks $[k+1..n]$ induce a penalty of $f_{k+1}(e+p_k)$, otherwise they would be rescheduled so as to reach this smaller penalty.

→ Or k is placed just after the block, that is with a tardiness of $p\Big([1..k-1]\Big) - e + p_k$. In this case, k induces an tardiness penalty of $w_k\ \Big(p\Big([1..k]\Big) - e\Big)$, and tasks $[k+1..n]$ induce a penalty of $f_{k+1}(e)$, since $\mathcal{S}$ is optimal.

This results in a dynamic programming with $O\Big(n\, p(J)\Big)$ states. Even if this number can be divided by two using other arguments proposed in [21], the algorithm is still pseudo-polynomial. Indeed, $p(J)$ is the time horizon, but not the problem size. For some instances, $p(J)$ can be exponential with respect to the problem size.

Remark 0.3

The Hall and Posner's algorithm can also be used for non-symmetric penalties provided that there exists a single order σ such that tasks are sorted both by non-increasing α-ratios and non-increasing β-ratios.

0.3.3 Algorithms for CDDP

Hoogeveen and Van de Velde [22] study CDDP in the case of symmetric earliness-tardiness penalties, *i.e.* $\forall j \in J,\ \alpha_j = \beta_j$. As previously, the unit penalties are then simply denoted $(w_j)_{j \in J}$. Moreover, they assume that processing times are integer.

• *Complexity results*

To establish the complexity of the problem, Hoogeveen and Van de Velde [22] consider the sub-case of task-independent unit penalties, *i.e.* $\forall j \in J, w_j = c$ for a constant $c \in \mathbb{R}$, with a restrictive due date, *i.e.* $d < p(J)$. They show that EOPP reduces to this problem. This proves that CDDP is NP-hard even if all unit penalties are equal.

More precisely, CDDP is weakly NP-hard as soon as unit penalties are symmetric, since it can be solved by a pseudo-polynomial algorithm proposed by Hoogeveen and Van de Velde [22]. This algorithm is based on the $\vee$-shaped d-or-left-block dominance property. The dominant set is decomposed into two subsets so that the schedules of the same subset share the same structure: on one hand the d-blocks, on the other hand the left-blocks. An optimal schedule is obtained by comparison of the best schedules within each subset. These optimal schedules are computed separately, using a different dynamic programming algorithm. Let us present both algorithms.

• *A pseudo-polynomial algorithm for the best d-block when unit penalties are symmetric*

Note that a $\vee$-shaped d-block is a V-shaped d-block as well. Therefore, as done for the unrestrictive case, one can consider only the d-blocks for which early (resp. tardy) tasks are scheduled in decreasing (resp. increasing) order of their indices, provided that tasks have been sorted by non-increasing w-ratio. Then in any considered d-block, for any $k \in [0..n]$, tasks $[1..k]$ form a d-block starting at time $d-e$, where $e = p\Big(E \cap [1..k]\Big) \in \mathbb{N}$.

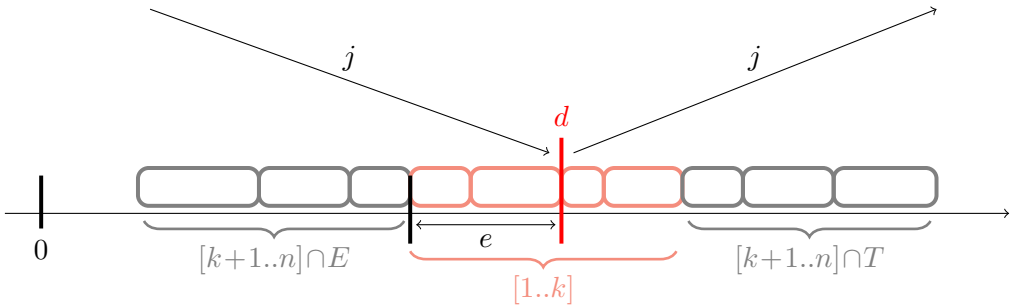


Figure 10: A d-block for the first Hoogeveen and Van De Velde's algorithm

For any $k \in [0..n]$ and any $e \in [0..d]$, let us denote by $\boldsymbol{\widetilde{f}_k(e)}$ the minimal penalty induced a d-block of tasks $[1..k]$ starting at time $d-e$. Note that we focus here on the penalty induced by central tasks, while f focuses on the penalty induced by external tasks. Since the starting time of an optimal d-block is not known a priori, the total penalty of an optimal d-block is given by $\min_{e \in [0..d]} \widetilde{f}_n(e)$.

Moreover, for any $e \in [0..d]$, $\widetilde{f}_0(e) = 0$ since $[1..0] = \emptyset$.

The value of $\widetilde{f}_k(e)$ for $e \in [0..d]$ and $k \in [1..n]$ can be computed by induction using the following recurrence relation.

$$\widetilde{f}_k(e) = \min\Big(\widetilde{f}_{k-1}(e-p_k) + w_k\,(e-p_k),\ \widetilde{f}_{k-1}(e) + w_k\,\Big(e + p\Big([1..k]\Big)\Big)\Big)$$

Indeed, in an optimal d-block $\mathcal{S}$ of tasks $[1..k]$ starting at time $d-e$, task k can only hold at two positions.

→ Either k is the first task of $\mathcal{S}$, its earliness is then $e-p_k$ and it induces an earliness penalty of $w_k\,(e-p_k)$. In this case, tasks $[1..k-1]$ form a d-block $\mathcal{S}'$ starting at time $e-p_k$. Moreover, by optimality of $\mathcal{S}$, $\mathcal{S}'$ is an optimal d-block starting at time $e-p_k$. We deduce that tasks $[1..k-1]$ in $\mathcal{S}$ induce a penalty of $\widetilde{f}_{k-1}(e-p_k)$.

→ Or k is the last task in $\mathcal{S}$, its tardiness is then $p\Big([1..k]\Big) - e$, and it induces a tardiness penalty of $w_k\,\Big(e + p\Big([1..k]\Big)\Big)$. In this case, tasks $[1..k-1]$ form a d-block $\mathcal{S}'$ starting at time e, by optimality of $\mathcal{S}$, we deduce that tasks they induce a penalty of $\widetilde{f}_{k-1}(e)$.

This results in a dynamic programming with $O\Big(n\,d\Big)$ states.

• *A pseudo-polynomial algorithm for the best left-block when unit penalties are symmetric*

In a left-block, one cannot say how the tasks are placed regarding to d. However, one knows that it starts at time 0 and completes at time $p(J)$. Therefore, the dynamic programming algorithm proposed here does not consider a central block but two blocks. One block to the left, starting at time 0, which is then a left-block, and completing before or at d, and one block to the right, completing at time $p(J)$, and starting after or at d.

Moreover, another difference with the d-blocks, is the $\vee$-shaped property instead of the V-shaped property. That implies that one cannot know which is the straddling task, it is not necessarily the

one with the largest ratio even if it is the first tardy task. Therefore, we compute successively for each $j_s \in J$ the best left-block in which j_s is the straddling task, and finally deduced the optimal left-block by comparison.
To explain how, let us set $j_s \in J$, and assume that tasks are sorted by *non-increasing* w-ratio (that is in the converse order to the previous one!). We consider only left-blocks where j_s is the straddling task and in which early (resp. tardy) tasks are scheduled in increasing (resp. decreasing) order of their indices. In addition, for any $(a,b) \in \mathbb{N}^2$, let us denote by $[a..b]^*$ the integer set $[a..b] \backslash \{j_s\}$. Hence, for any $k \in [0..n]$, tasks $[1..k]^*$ are divided into two blocks, one starting at time 0 and completing at time $l = p\Big([1..k]^* \cap E\Big) \leqslant d$, and another starting at time $p(J) - \Big(p([1..k]^*) - l\Big) = p\Big([k+1..n] \cup \{j_s\}\Big) + l$ and completing at time $p(J)$. Tasks $[k+1..n]^* \cup \{j_s\}$ form a block in the middle.

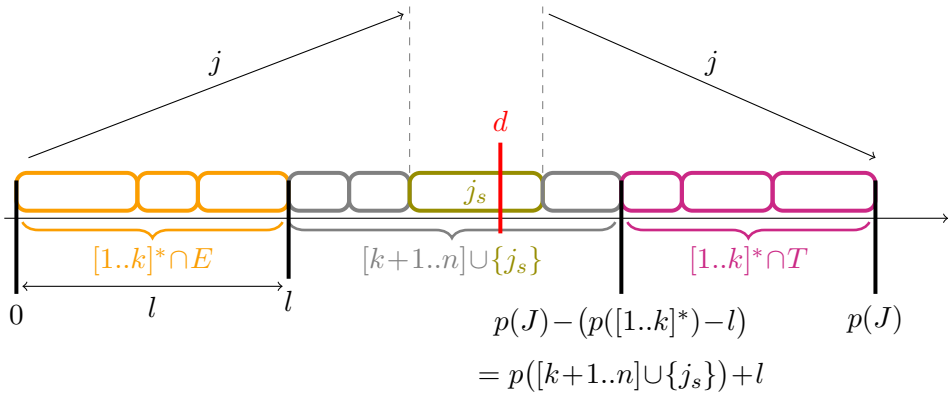


Figure 11: A left-block for the second Hoogeveen and Van De Velde's algorithm

For any $k \in [0..n]$ and any $l \in [-p(J)..d]$, let us denote by $\boldsymbol{\widehat{f}_k(l)}$ the minimal penalty induced by a schedule of tasks $[1..k]$ in the two time intervals $[0,l]$ and $[p\Big([k+1..n] \cup \{j_s\}\Big) + l, p(J)]$. If no such schedule exists, $\widehat{f}_k(l) = \infty$. Therefore, $\widehat{f}_k(l) = \infty$ when $l > p([1..k]^*)$, in particular when $k = 0$ and $l > 0$. Conversely, $\widehat{f}_0(0) = 0$.
Since the starting time of the straddling task j_s is not known a priori, the total penalty of an optimal left-block in which j_s is the straddling task is given by $\min_{l \in [d - p_{j_s}..d]} \widehat{f}_n(l)$.

The value of $\widehat{f}_k(l)$ for $k \in [1..n]$ and $l \in [0..d]$ can be computed by induction using the following recurrence relation.

$$\widehat{f}_k(l) = \begin{cases} \widehat{f}_{k-1}(l) & \text{if } k = j_s \\ \min\Big(\widehat{f}_{k-1}(l - p_k) + w_k\,(d - l),\ \widehat{f}_{k-1}(l) + w_k\,\Big(p\Big([k+1..n] \cup \{j_s\}\Big) + l + p_k - d\Big)\Big) & \text{otherwise} \end{cases}$$

Indeed, in an optimal schedule $\mathcal{S}$ of tasks $[1..k]$ in $[0,l]$ and $[p\Big([k+1..n] \cup \{j_s\}\Big) + l, p(J)]$, task k can only hold at two positions.

- → Either k is the last task in the left interval, that is k completes at time l and then induces an earliness penalty of $w_k\,(d - l)$. In this case, tasks $[1..k-1]$ are placed in the two intervals $[0..l - p_k]$ and $[r, p(J)]$ where $r = p\Big([k+1..n] \cup \{j_s\}\Big) + l = p\Big([k..n] \cup \{j_s\}\Big) + (l - p_k)$. By optimality of $\mathcal{S}$, these tasks $[1..k-1]$ induce a penalty of $\widehat{f}_{k-1}(l - p_k)$.

- → Or k is first task in the right interval, that is k completes at time $r = p\Big([k+1..n] \cup \{j_s\}\Big) + l + p_k$. In this case, tasks $[1..k-1]$ are placed in the two intervals $[0..l]$ and $[r, p(J)]$. Since $r = p\Big([k..n] \cup \{j_s\}\Big) + l$, by optimality of $\mathcal{S}$, we deduce that tasks $[1..k-1]$ induce a penalty of $\widehat{f}_{k-1}(l)$.

This results in a dynamic programming with $O\Big(n\,p(J)\Big)$ states, which as to be repeat n times to consider all possible straddling tasks.

Remark 0.4

Like the Hall and Posner's algorithm, both Hoogeveen and Van De Velde algorithms can be used for non-symmetric penalties provided that there exists a single order σ such that tasks are sorted both by non-increasing α-ratios and non-increasing β-ratios.

• *A heuristic for CDDP, along with a benchmark*

Biskup and Feldmann [9] propose two heuristics algorithm for CDDP, without any assumptions on the unit penalties.

In contrast with the exact algorithms proposed above, these heuristic algorithms are not exactly based on dominance properties since they consider onlyV-shaped d-blocks, which do not form a dominant set for an arbitrary due date. As we will explain in Section 0.5, the total penalty of such a schedule can be computed from the partition between early and tardy tasks only. Therefore, both heuristic algorithms proposed by Biskup and Feldmann consist in building an early-tardy partition (E, T).

The first heuristic algorithm starts with $E = \emptyset$ and $T = J$. At each step, the penalty variation induced by moving j from T to E. is compute for each task $j \in T$ such that $p_j \leqslant d - p(E)$. If no such task induces a penalty reduction, the algorithms stops and returns the V-shaped d-block associated with the partition (E, T). Otherwise, the task inducing the largest reduction is removed from J and added to E.

The second heuristic algorithm starts with $E = \emptyset$ and $T = \emptyset$. Tasks are considered one by one by decreasing β-ratio. While $|E| < \lfloor n/2 \rfloor$, if the considered task j satisfies $p_j \leqslant d - p(E)$, then j is added to E, otherwise j is added to T. When $\lfloor n/2 \rfloor$ tasks have been added to E, all the remaining tasks are added in T, However, these tasks are considered by decreasing β-ratio once again in order to decide if they stay in T or not: a task is moved from T to E if it induces a penalty reduction.

These algorithms provide a feasible schedule which is not necessary optimal, and no guarantee is provided to bound the ratio of its penalty over the penalty of an optimal schedule. To assess the efficiency of their algorithms, the authors provide a benchmark with both restrictive and unrestrictive instances, with arbitrary penalties.
Both algorithms succeed in providing a solution within few seconds, even for large instances ($n = 1000$). However, authors can only evaluate the quality of these solutions for small instances ($n = 10$), since they have no exact methods to solve large instances. For these small instances, the penalty of the provided solutions is close to the optimal value for a restrictive due date (the average gap over the ten instances is 6.7%) and very closed when the due date is unrestrictive (9 over the 10 instances are solved to optimality).

In order to compare the heuristic that we proposed in Section 6.3 for UCDDP, we implement the more promising[6] of the Biskup and Feldmann's algorithms. Therefore, we are able to evaluate the quality of the solutions provided by their heuristic for lager size instances ($n = 200$): they are very good since the average gap for unretsrictive instances is 0.1%, *Cf.* Table 6.3 on page 181. However, these solutions can be improved by a local search algorithm proposed in Section 6.3.

• *Algorithms designed for more general problems*

A generalization of CDDP is the single machine scheduling problem with **distinct due dates**, in which each task $j \in J$ has its own due date d_j. The earliness (resp. tardiness) of j is then $E_j = [d_j - C_j]^+$

[6] According to the authors themselves.

(resp. $T_j = [C_j - d_j]^+$). The algorithms designed to solve this problem can be used to solve CDDP. Among them, one can cite the Branch-and-Bound algorithm proposed by Sourd [41], which have been tested on the Biskup and Feldmann benchmark. This algorithm is able to exactly solve instances up to size 1000 within 1400 seconds at most. To the best of our knowledge, this algorithm is the most efficient for solving to optimality UCDDP and CDDP.

Araki, Fujikuma and Tanaka [42] propose a Successive Sublimation Dynamic Programming (SSDP) algorithm to solve a general single machine scheduling problem, where the penalty of a schedule can be written as $\sum_{j \in J} f_j(C_j)$, where, for any task j, f_j is an integer-valued function. This kind of functions covers all the cases where the total penalty of a schedule is the sum of the penalty of each task, and the latter only depends on the task completion time. In particular, that covers function $f_{\alpha,\beta,d}$. However, no experimental results are given for common due date problems but only for distinct due date problems.

The complexity results given in the last three sections are contextualized in the next section, in which we propose an overview of the complexity results known for problems related to UCDDP and CDDP.

0.3.4 Due date related scheduling problems

On page 32, we propose a cartography of single machine scheduling problems related to CDDP. More precisely, all the problems represented can be derived from CDDP by changing assumptions on the due date, the processing times, and/or the unit penalties, or by adding some assumptions.
Note that, in spite of their name, some assumptions induce no restriction on the instances. For example, "restrictive due date" is only written by opposition to "unrestrictive due date", and means that the due date is not necessarily unrestrictive, *i.e.* d can be arbitrarily chosen smaller or greater than $p(J)$. To avoid misunderstanding, the following table specifies the different assumptions appearing on the cartography.

- Assumptions on the due date:
  - distinct due dates means that each task j has its own due date d_j. These due dates are not necessarily distinct.
  - **common** due date means that $\forall j \in J$, $d_j = d$.
  - restrictive for a common due date d means that d can be small.
  - unrestrictive for a common due date d means that $d \geqslant p(J)$.
- Assumptions on the unit earliness and tardiness penalties:
  - **symmetric** penalties means that earliness and tardiness are penalized in the same way, *i.e.* $\forall j \in J$, $\alpha_j = \beta_j$. The unit penalty of task j is then denoted by w_j.
  - arbitrary penalties conversely means that the earliness and the tardiness of each task j are penalized according to different parameters α_j ans β_j. It is not necessary that $\alpha_j \neq \beta_j$.
  - **task-independent** penalties means that $\forall j \in J$, $\alpha_j = \alpha$ and $\beta_j = \beta$ for given α and β
  - task-dependent penalties means that each task j has its own unit penalties α_j and β_j. These penalties are not necessarily distinct from a task to another.
- Assumptions on the processing times:
  - $p_i = p$ means that all the tasks have the same processing time, *i.e.* $\forall j \in J$, $p_j = p$
  - $p_j = w_j$ for symmetric penalties $(w_j)_{j \in J}$ means that $\forall j \in J$, $p_j = w_j$
- Additional assumption:

– fixed sequence means that a task sequence is given in the instance,

Each problem is represented by a rectangle divided in two parts: the left part gives its definition while the right part gives its complexity, if it is known. In addition, some explanations are given in gray on the right part, either to justify why the problem is NP-hard, or to specify its complexity when the problem is in P.

• *How to read this cartography*

The idea is to start from a problem, like UCDDP, and then browse the cartography going from a problem to another following the arrows. Indeed, an arrow is represented from a problem to another if only one assumption is added or removed.
However, problems can also be considered by batch of four problems differing from each other only on assumptions about unit penalties. Such batches are highlighted using mauve ellipses in background.

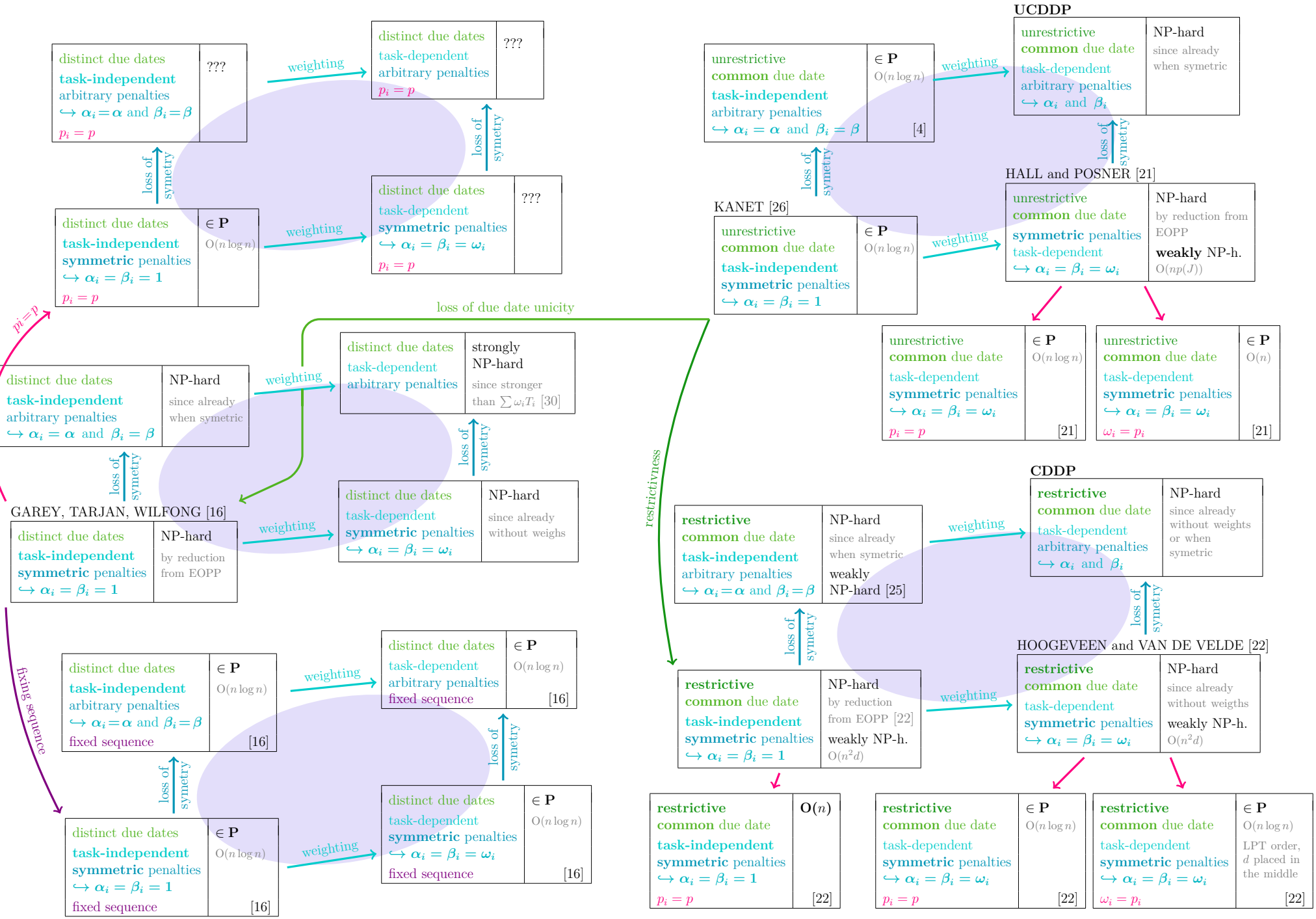
• *Changes making a problem hard or conversely easy*

Starting from the problem studied by Kanet (unrestrictive common due date, all unit penalties being equal), which is solvable in $O(n \log n)$, three assumptions makes the problem NP-hard when they are relaxed.

- By relaxing the due date unrestrictiveness, the problem becomes weakly NP-hard as shown by Hoogeveen and Van De Velde [22].
- By relaxing the due date unicity, the problem becomes NP-hard as shown by Garey, Tarjan, and Wilfong [16].
- By relaxing the task-independence of unit penalties, the problem becomes weakly NP-hard as shown by Hall and Posner [21].

Conversely, NP-hard problems can become solvable in polynomial time by assuming that all processing times or all α and β-ratios are equal, or when a task sequence is given.

The cartography given on the next page concludes our state of the art for the scheduling field. In the next section, we introduce the linear programming—from what is an LP to what is a Branch-and-Cut algorithm—and present some linear formulations for single machine scheduling problems.


distinct due dates
task-independent
arbitrary penalties
↪ $\alpha_i = \alpha$ and $\beta_i = \beta$
$p_i = p$
???
weighting
distinct due dates
task-dependent
arbitrary penalties
$p_i = p$
???
loss of symetry
distinct due dates
task-independent
symmetric penalties
↪ $\alpha_i = \beta_i = 1$
$p_i = p$
∈ P
$O(n \log n)$
weighting
distinct due dates
task-dependent
symmetric penalties
↪ $\alpha_i = \beta_i = \omega_i$
$p_i = p$
???
$p_i = p$
loss of due date unicity
distinct due dates
task-independent
arbitrary penalties
↪ $\alpha_i = \alpha$ and $\beta_i = \beta$
NP-hard
since already when symetric
weighting
distinct due dates
task-dependent
arbitrary penalties
strongly NP-hard
since stronger than $\sum \omega_i T_i$ [30]
loss of symetry
GAREY, TARJAN, WILFONG [16]
distinct due dates
task-independent
symmetric penalties
↪ $\alpha_i = \beta_i = 1$
NP-hard
by reduction from EOPP
weighting
distinct due dates
task-dependent
symmetric penalties
↪ $\alpha_i = \beta_i = \omega_i$
NP-hard
since already without weighs
fixing sequence
distinct due dates
task-independent
arbitrary penalties
↪ $\alpha_i = \alpha$ and $\beta_i = \beta$
fixed sequence
[16]
∈ P
$O(n \log n)$
weighting
distinct due dates
task-dependent
arbitrary penalties
fixed sequence
[16]
∈ P
$O(n \log n)$
loss of symetry
distinct due dates
task-independent
symmetric penalties
↪ $\alpha_i = \beta_i = 1$
fixed sequence
[16]
∈ P
$O(n \log n)$
weighting
distinct due dates
task-dependent
symmetric penalties
↪ $\alpha_i = \beta_i = \omega_i$
fixed sequence
[16]
∈ P
$O(n \log n)$
restrictivness
unrestrictive
common due date
task-independent
arbitrary penalties
↪ $\alpha_i = \alpha$ and $\beta_i = \beta$
∈ P
$O(n \log n)$
[4]
weighting
UCDDP
unrestrictive
common due date
task-dependent
arbitrary penalties
↪ α_i and β_i
NP-hard
since already when symetric
loss of symetry
KANET [26]
unrestrictive
common due date
task-independent
symmetric penalties
↪ $\alpha_i = \beta_i = 1$
∈ P
$O(n \log n)$
weighting
HALL and POSNER [21]
unrestrictive
common due date
symmetric penalties
task-dependent
↪ $\alpha_i = \beta_i = \omega_i$
NP-hard
by reduction from EOPP
weakly NP-h.
$O(np(J))$
unrestrictive
common due date
task-dependent
symmetric penalties
↪ $\alpha_i = \beta_i = \omega_i$
$p_i = p$
∈ P
$O(n \log n)$
[21]
unrestrictive
common due date
task-dependent
symmetric penalties
↪ $\alpha_i = \beta_i = \omega_i$
$\omega_i = p_i$
∈ P
$O(n)$
[21]
restrictive
common due date
task-independent
arbitrary penalties
↪ $\alpha_i = \alpha$ and $\beta_i = \beta$
NP-hard
since already when symetric
weakly NP-hard [25]
weighting
CDDP
restrictive
common due date
task-dependent
arbitrary penalties
↪ α_i and β_i
NP-hard
since already without weights or when symetric
loss of symetry
restrictive
common due date
task-independent
symmetric penalties
↪ $\alpha_i = \beta_i = 1$
NP-hard
by reduction from EOPP [22]
weakly NP-h.
$O(n^2 d)$
weighting
HOOGEVEEN and VAN DE VELDE [22]
restrictive
common due date
task-dependent
symmetric penalties
↪ $\alpha_i = \beta_i = \omega_i$
NP-hard
since already without weigths
weakly NP-h.
$O(n^2 d)$
restrictive
common due date
task-independent
symmetric penalties
↪ $\alpha_i = \beta_i = 1$
$p_i = p$
O(n)
[22]
restrictive
common due date
task-dependent
symmetric penalties
↪ $\alpha_i = \beta_i = \omega_i$
$p_i = p$
∈ P
$O(n \log n)$
[22]
restrictive
common due date
task-dependent
symmetric penalties
↪ $\alpha_i = \beta_i = \omega_i$
$\omega_i = p_i$
∈ P
$O(n \log n)$
LPT order, d placed in the middle
[22]

0.4 MIP formulations for scheduling problems

In order to present MIP formulations, we first define what is a linear formulation for a minimization problem. Of course they can be transposed to a maximization problem. The convex analysis definitions used (polyhedron, convex...) are recalled in appendix (*Cf.* page 203).

0.4.1 Linear formulations and polyhedra

Let $d \in \mathbb{N}^*$. A **linear inequality**[7] in dimension d is defined by a normal vector $a \in \mathbb{R}^d \setminus \{0\}$, a real number $\alpha \in \mathbb{R}$, and an inequality symbol, *i.e.* $\leqslant$ or $\geqslant$. A vector $x \in \mathbb{R}^d$ **satisfies** the inequality if $a \cdot x \leqslant \alpha$ (resp. $a \cdot x \geqslant \alpha$). Otherwise, one say that x **violates** the inequality. An inequality is **valid** for a set $S \subseteq \mathbb{R}^d$ if all the points of S satisfies it. A **linear program** (LP) in dimension d is defined by a linear objective function f from $\mathbb{R}^d$ to $\mathbb{R}$ and a finite set of linear inequalities, indexed by $\mathcal{I}$. Such an LP can be written as follows for a cost vector $c \in \mathbb{R}^d$ and a family $(a^i, \alpha^i)_{i \in \mathcal{I}} \in \mathbb{R}^{\mathcal{I}}$.

$$\min\{\, c \cdot x \;\; | \forall i \in \mathcal{I},\, a^i \cdot x \leqslant \alpha^i \}$$

Such an LP can be solved in polynomial time (with respect to the LP size) using the interior-point method proposed by Karmarkar in 1984 [27]. Moreover, several LP-solvers are available to solve such problems. Even though they are based on the simplex algorithm proposed by Dantzig in 1947, which can require exponentially-many computation steps, these solvers are efficient in practice, and broadly used.

From a geometrical point of view, the solution set of an LP is a polyhedron, since each inequality is satisfied by an half-space of $\mathbb{R}^d$. Let us denote by P this polyhedron and assume that $P \neq \emptyset$. Since P is a closed set, function f, which is a linear and then continuous, admits a minimum on P. Note that several points in P can reach this minimum. Actually, If there exist two minimizers $x \neq y$ in P, then the whole segment $[x, y]$ is in P (by convexity of P), and any point in $[x, y]$ is also a minimizer (by linearity of f). An important particularity of polyhedra among closed convex sets, is that they can be finitely generated, in the sense of Klee's Theorem (*Cf.* page 208). Indeed, a polyhedron has finitely-many extreme points, and at least one of them reaches the minimum. Based on this remark, the simplex algorithm browses the extreme points of the polyhedron, guided by the objective function. The minimizer provided by this algorithm is then an extreme point of the polyhedron.

LP can model a wide class of practical problems. However, a priori (i.e. without further investigations on the polyhedron) variables have to model continuous quantities, which is restrictive. Let us introduce a small example to illustrate this remark.

0.4.2 Continuous variables *vs.* integer variables, or bulk *vs.* packets

Imagine that I have a 10€ voucher to buy flour directly to a mill, where green lentil, chickpea, spelt, and wheat flours are sold from bulk. I want to use my voucher, that is spend at least 10€ and to buy at least 500g of each kind of flour, but I also want to minimize my shopping weight. Moreover, I use only chickpea or green lentil flour mixed with twice as much wheat or spelt flour. The best solution for me, *i.e.* is the best weight of each flour $(w_{gl}, w_{cp}, w_s, w_w)$ that I can buy, can be found by solving the following LP, where c_{gl}, c_{cp}, c_s, c_w denote the price per kg of green lentil, chickpea, spelt, and wheat flour respectively.

[7] Note that a *linear* inequality $a \cdot x \leqslant \alpha$ corresponds to an half-space delimited by an *affine* hyperplane, since the right member α is not necessarily zero.

$$\min \left\{ w_{gl} + w_{cp} + w_s + w_w \;\middle|\; \begin{array}{l} (w_{gl}, w_{cp}, w_s, w_w) \in \mathbb{R}^4 \\ c_{gl}\, w_{gl} + c_{cp}\, w_{cp} + c_s\, w_s + c_w\, w_w \geqslant 10 \\ w_{gl} \geqslant 0.5,\ w_{cp} \geqslant 0.5,\ w_s \geqslant 0.5, \text{ and } w_w \geqslant 0.5 \\ w_s + w_w \geqslant 2\,(w_{gl} + w_{cp}) \end{array} \right\}$$

Note that this problem can be formulated as an LP since the amount of each floor can be represented by a continuous variable. If you assume, in the same example, that flours are not sold from bulk, but in 5kg-packets for wheat and in 1kg-packet for the others, the problem is modeled by integer variables, n_{gl}, n_{cp}, n_s, n_w representing the number of packets for each kind of flour. The problem is then formulated as follows, where $c^1_{gl}, c^1_{cp}, c^1_s$, and c^1_w denote the packet prices.

$$\min \left\{ n_{gl} + n_{cp} + n_s + 5\, n_w \;\middle|\; \begin{array}{l} (n_{gl}, n_{cp}, n_s, n_w) \in \mathbb{N}^4 \\ c^1_{gl}\, n_{gl} + c^1_{cp}\, n_{cp} + c^1_s\, n_s + c^1_w\, n_w \geqslant 10 \\ n_{gl} \geqslant 0.5,\ n_{cp} \geqslant 0.5,\ n_s \geqslant 0.5, \text{ and } 5\, n_w \geqslant 0.5 \\ n_s + 5\, n_w \geqslant 2\,(n_{gl} + n_{cp}) \end{array} \right\}$$

The solution set of this formulation is no longer a polyhedron, but the set of the integer points of a polyhedron. Such a program is called an **integer program** (IP) .

Assuming that the wheat flour only is sold from bulk and no longer in 5kg-packet, the problem is then formulated by the following **mixed-integer program** (MIP) where continuous and integer variables coexist.

$$\min \left\{ n_{gl} + n_{cp} + n_s + w_w \;\middle|\; \begin{array}{l} (n_{gl}, n_{cp}, n_s, w_w) \in \mathbb{N}^3 \times \mathbb{R} \\ c^1_{gl}\, n_{gl} + c^1_{cp}\, n_{cp} + c^1_s\, n_s + c_w\, w_w \geqslant 10 \\ n_{gl} \geqslant 0.5,\ n_{cp} \geqslant 0.5,\ n_s \geqslant 0.5, \text{ and } w_w \geqslant 0.5 \\ n_s + w_w \geqslant 2\,(n_{gl} + n_{cp}) \end{array} \right\}$$

In the sequel, we will use MIP to design both IP or MIP. Conversely, LP will only designate programs without integrity constraints. Note that a MIP can be turned into an LP by relaxing (*i.e.* removing) its integrity constraints. The LP is then called the **linear relaxation** of the MIP. In the above example, the first LP formulation is the linear relaxation of the last formulation. Indeed, the objective function and the linear inequalities are the same[8] but $(n_{gl}, n_{cp}, n_s) \in \mathbb{N}^3$ is relaxed into $(n_{gl}, n_{cp}, n_s) \in \mathbb{R}^3$.

MIP can model a very broad range of problems, which includes NP-hard problems, then, in contrast with LP, MIP are not solvable in polynomial time in general. In some cases, it can be shown that the MIP has the same value as its linear relaxation, for example when all the extreme points of the polyhedron are integer points. If so, it suffices to solve the linear relaxation, which is easy if the formulation is not too big. To be precise, for a given problem, a formulation is said **compact** if the number of variables and the number of inequalities are bounded by a polynomial function of the instance size. If a problem reduces to solve a compact LP, then it can be solved in polynomial time.

We do not develop such cases in the following, and conversely focus of how to solve MIP where integrity constraints cannot be omitted, and then LP holding exponentially-many inequalities.

[8] They are the same, as soon as variables n_{gl}, n_{cp}, and n_s are renamed into w_{gl}, w_{cp}, and w_s respectively.

0.4.3 Branch-and-Bound algorithm to solve MIP

Let us present the **Branch-and-Bound** method in general, for an arbitrary minimization problem $\mathcal{P}^0$. Let us denote by z^* its optimal value. Of course the following results can be transposed to a maximization problem. A more developed introduction to Branch-and-Bound algorithms can be found in [38].

• *Branch-and-Bound algorithms in general*

A Branch-and-Bound algorithm requires a sub-algorithm, called **bounding function**, able to provide a lower bound for $\mathcal{P}^0$ and for **reinforcements** of $\mathcal{P}^0$, that are problems obtained from $\mathcal{P}^0$ by adding some constraints. Let us denote this sub-algorithm by LB. and by z^* the optimal value of $\mathcal{P}^0$. The more a problem $\widehat{\mathcal{P}}$ is reinforced compared to $\mathcal{P}^0$, the larger is its optimal value $\widehat{z}$, and hopefully the larger is $\texttt{LB}(\widehat{\mathcal{P}})$. Note that $\texttt{LB}(\widehat{\mathcal{P}})$ is not a lower bound on z^* in general. However, if the solution subsets of a family of reinforcements cover the solution set of $\mathcal{P}^0$, then the minimal lower bound of these reinforcements is a lower bound on z^*. To decide how to divide the solution set, that is to chose which reinforcements of a problem to consider, the Branch-and-Bound algorithm also requires a **branching rule**.

A Branch-and-Bound algorithm handles both a lower and an upper bound on z^*. The aim is to reduce the gap between these bounds so that only one possible value remains in the interval: z^*. The lower bound is improved by:

→ applying the branching rule to generate problem reinforcements,

→ applying the bounding function to each reinforcements,

→ deducing a new lower bound by taking the minimal lower bound,

→ recursively applying this method to improve the lower bounds.

The execution of a Branch-and-Bound algorithm can be represented by a decision tree, called the **branching tree**. Each node represents a reinforcement of $\mathcal{P}^0$. The root node is $\mathcal{P}^0$ itself. The children of a node $\mathcal{P}$ are reinforcements of $\mathcal{P}$, obtained by making additional assumptions, called **branching decisions**. The solution sets of the children must cover the solution set of $\mathcal{P}$. Thanks to this, the minimal value of the child lower bounds is a lower bound for $\mathcal{P}$, hopefully better, *i.e.* larger, than the one provided by $\texttt{LB}(\mathcal{P})$.

In addition, a Branch-and-Bound algorithm requires a way to choose which node is explored at each step. Whichever strategy is chosen (depth first, best first...), there exists two kinds of nodes $\mathcal{P}$ that do not need to be explored.

(i) If an optimal solution x is known for $\mathcal{P}$, then its value is an exact lower bound for $\mathcal{P}$, there is no way to improve this bound by considering reinforcements. Such a solution can be found, by any chance, when applying LB, or by applying on purpose a heuristic method to solve $\mathcal{P}$, or even by applying an exact method if $\mathcal{P}$ is sufficiently reinforced to be easy to solve. Moreover, x is also a solution for $\mathcal{P}^0$ then its value gives an upper bound on z^*.

(ii) If $\texttt{LB}(\mathcal{P})$ is larger than the best currently known upper bound on z^*, there is no need to explore $\mathcal{P}$. Indeed, this node will never have the minimal lower bound, and will never be used as a lower bound for z^*, since improving a lower bound makes it only largest. In other words, the solution set of $\mathcal{P}$ corresponds to solutions having a too large value according to the objective function, and it will be worse for any reinforcement of $\mathcal{P}$. Note that this case particularly occurs when $\mathcal{P}$ solution set is empty, and the lower bound for $\mathcal{P}$ is $+\infty$.

When the potential sub-tree rooted in such a node is not explored, we say that we **prune** the branching tree. Pruning is interesting since it allows to save node exploration.

• *Branch-and-Bound algorithms for solving MIP*

Let us now present how Branch-and-Bound algorithm components are instantiated for MIP exact solving.

- A branching decision consists in adding a linear inequality involving only one integer variable, which is then called the **branching variable**, *i.e.* an inequality like $x_i \leqslant a$ or $x_i \geqslant a$. Therefore, the reinforcements of the initial MIP are also MIP, or even LP.

- The bounding function consists in solving the linear relaxation of the MIP related to the node. Therefore, reinforcing a problem necessarily leads to a no smaller lower bound. Moreover, if the solution found by solving the linear relaxation is integer, its value is an exact lower bound for this node, and also an upper bound on z^*. Otherwise, a rounding procedure may be applied in order to produce a feasible solution and then an upper bound on z^*.

- The branching rule is based on the solution $\widetilde{x}$ of the linear relaxation. The decision variable x_i is chosen among the integer variables that have the largest fractional part in $\widetilde{x}$. Two branching decisions are then considered: $x_i \leqslant \lfloor \widetilde{x}_i \rfloor$ and $x_i \geqslant \lceil \widetilde{x}_i \rceil$. The corresponding solution subsets cover all the feasible solutions, since the value of x_i has to be integer.
  Note that, in the case of binary variables, the branching decisions are necessary $x_i \leqslant 0$ and $x_i \geqslant 1$, which are equivalent to $x_i = 0$ and $x_i = 1$. In other words, each branching step consists in fixing one variable. If the n integer variables are n binary variables, a node of depth n admits only one solution, which is an integer solution and provides an upper bound.

The efficiency of a Branch-and-Bound algorithm depends in general of the quality of the lower bounds provided by the bounding function. Indeed, the tighter they are, the more we can prune a node of type (ii). In the MIP solving framework, the quality of the lower bounds directly depends on the quality of the initial formulation. In addition, a best formulation can also allow to obtain an integer solution when solving the linear relaxation, which corresponds to a node of type (i) and potentially improves the upper bound on z^*.

Therefore, when a problem is formulated as a MIP, we are interested in its **linear relaxation value**, *i.e.* the optimal value of the linear relaxation. Roughly speaking, a formulation with a linear relaxation value near to the optimal value is a formulation for which the Branch-and-Bound solving will requires to explore few nodes. However, in practice, adding linear inequalities to improve the linear relaxation value can slow down the linear relaxation solving step at each node. Therefore, it does not necessary reduce the global computation time required by the Branch-and-Bound algorithm. Such cases will be discussed in Section 3.5.

0.4.4 Branch-and-Cut algorithm

Even if LP are solvable in polynomial time, when the number of inequalities becomes too large, solving an LP can be very long, or at least too long to be repeated at each node of a Branch-and-Bound algorithm. A common way to manage inequalities when they are too numerous, is to add them progressively in the LP given to the solver, according to a separation algorithm. The aim is to add only a small part of the inequalities, while ensuring that the solution finally obtained satisfies them all. We present first what is a separation algorithm, and then explain how it can be used in a **cutting plane based algorithm** to solve an LP.

• *Separation problem and separation algorithm*

Let us consider a family of inequalities $a^i \cdot x \leqslant \alpha^i$ indexed by a finite set $\mathcal{I}$. The **separation problem** associated with this inequality family is to decide for a point in $x \in \mathbb{R}^d$ whether it satisfies all the inequalities of the family, AND, if not, to provide a violated inequality.

SEPARATION	Input: $x \in \mathbb{R}^d$
	Output: "yes" if $\forall i \in \mathcal{I}$, $a^i \cdot x \leqslant \alpha^i$ an index $i \in \mathcal{I}$ such that $a^i \cdot x > \alpha^i$ otherwise

Note that it is not a simple decision problem, since in the negative case, a violated inequality is required as a proof. A **separation algorithm** is an algorithm that solves such separation problem. A separation algorithm can provide several indices of unsatisfied inequalities, or it can provide the index of a most violated inequality, *i.e.* maximizing $a^i \cdot x - \alpha^i$ One uses also "separation algorithm" to designate an algorithm that solves the sub-problem where x is integer. Indeed, the separation problem can be easier to solve for integer points, and in this case, it is advantageous to use a special algorithm for these points.

• *Cutting plane based algorithm*

Let us consider the following LP.

$$\min \left\{ c \cdot x \;\middle|\; \begin{array}{l} x \in \mathbb{R}^d \\ \forall i \in \mathcal{I},\ a^i \cdot x \leqslant \alpha^i \\ \forall j \in \mathcal{J},\ b^j \cdot x \leqslant \beta^j \end{array} \right\}$$

Let us assume that only the first inequality family has to be separated, *i.e.* inequalities $a^i \cdot x \leqslant \alpha^i$ for $i \in \mathcal{I}$, which will be called a-inequalities. We introduce the polyhedron defined by the other inequality family.

$$\widetilde{P} = \left\{ x \in \mathbb{R}^d \;\middle|\; \forall j \in \mathcal{J},\ b^j \cdot x \leqslant \beta^j \right\}$$

For a given separation algorithm for the a-inequalities, the **cutting plane based algorithm** is the following.

0. Start with $\widetilde{\mathcal{I}} = \emptyset$, that is with no a-inequality.
1. Solve the LP and record its optimal solution $\widetilde{x}$, *i.e.* $\widetilde{x} = \arg\min \left\{ c \cdot x \;\middle|\; \begin{array}{l} x \in \widetilde{P} \\ \forall i \in \widetilde{\mathcal{I}},\ a^i \cdot x \leqslant \alpha^i \end{array} \right\}$
2. Launch the separation algorithm on $\widetilde{x}$
3. If all the a-inequalities are satisfied, return $\widetilde{x}$.
   Otherwise, add to $\widetilde{\mathcal{I}}$ the index provided by the separation algorithm, and go back to step 1.

An important result about the complexity of cutting plane based algorithms is due to Grötschel, Lovász and Schrijver [19]. They show that a cutting plane based algorithm can be solved in polynomial time if and only if the related separation algorithm is solvable in polynomial time.

A MIP can combine both difficulties: an exponential number of inequalities along with integrity constraints. In this case, we use a **Branch-and-Cut algorithm**, which is actually a Branch-and-Bound algorithm in which the algorithm used to solve each LP is a cutting plane based algorithm.

0.4.5 Linear formulations for scheduling problems

Many scheduling problems are formulated by LP, IP or MIP, even in cases where they are not solved using linear programming techniques. The nature of these linear formulations greatly depends on the variables used to described a schedule. We briefly presents some of them in the case of single machine scheduling problems. Within more general framework, polyhedral approaches using these variable families are presented in a report of Queyranne and Schulz [35].

- *Completion time variables* $(C_j)_{j\in J}$

These variables are continuous variables representing the completion times as they are defined page 10. Without speaking about linear formulation, most of the single machine scheduling problems are first formulated using the completion times. Therefore, they are called natural date variables in [35].

As presented on page 51, the fundamental non-overlapping constraints (1.1) are not linear inequalities for completion time variables. However, Queyranne proposes in [34] a way to ensure task non-overlapping with linear inequalities using only completion time variables. This result is further discussed in Section 1.1 as we propose to extend the idea in order to provide linear formulations for common due date scheduling problems based on natural variables, which are similar to completion time variables.

To the best of our knowledge, no linear formulation using completion time variables has been proposed for common due date problems.

- *Time-indexed variables* $(x_{j,t})_{(j,t)\in J\times[1..T]}$

In case of integer processing times, the task completion times in a block starting at an integer time are all integers. Therefore, using dominance properties, one can assume that task completion times belong to a range $[1..T]$, where T is called the **time horizon**. Then a schedule can be described by the so-called **time-indexed variables**, $(x_{j,t})_{(j,t)\in J\times[1..T]}$, which indicate for each task j if it completes at time t. In contrast with the previous variables, these ones are integer (and even binary). Moreover, they can model a lot of constraints without additional variables, then they generally result in IP formulations, and not in MIP formulations. Such IP formulations often offer a good linear relaxation value, but their size is sensitive to the time horizon, and when it becomes too large, the number of variables skyrockets.

The fundamental non-overlapping constraints can be formulated using time-indexed variables by linear inequalities (2) presented below.

Even if the Sourd's solving approach for CDDP in [41] does not fall under linear programming, a time-indexed formulation for CDDP is proposed. Let us present below this formulation, that we solve in Section 2.3.

Let us assume that the due date and the processing times are integer, *i.e.* $d\in\mathbb{N}$ and $(p_j)_{j\in J}\in\mathbb{N}^J$. The task completion times for a d-block are then necessary integers between $T_{\min}=d-p(J)$ and $T_{\max}=d+p(J)$. For the sake of brevity, we rather use $T_{\min}=1$ and denote simply by T the $T_{\max}$. For any $t\in[1...T]$ and any $j\in J$, the penalty induced by the task j if it completes at time t can be

expressed as follows.

$$c_{j,t} = \max\Big(\alpha_i(d-t),\, \beta_i(t-d)\Big) = \alpha_i[d-t]^+ + \beta_i[t-d]^+$$

Moreover, the following inequalities ensure that each task completes exactly at one time and tasks do not overlap.

$$\forall j \in J,\ \sum_{t=1}^{T} x_{j,t} = 1 \tag{1}$$

$$\forall t \in [1..T],\ \sum_{j\in J} \sum_{s=t}^{\min(t+p_j-1,\,T)} x_{j,s} \leqslant 1 \tag{2}$$

Let us introduce the polyhedron defined by these inequalities, along with the set of its integer points.

$$\boldsymbol{P^{TI}} = \Big\{(x_{j,t})_{(j,t)\in J\times[1..T]} \in \mathbb{R}^{J\times[1..T]} \;\Big|\; (1-2) \text{ are satisfied}\Big\} \qquad \mathrm{int}_{\boldsymbol{x}}(\boldsymbol{P^{TI}}) = P^{TI} \cap \{0,1\}^{J\times[1..T]}$$

The time-indexed formulation for CDDP is then the following.

$$F_{\mathrm{TI}} : \quad \min_{x\in \mathrm{int}_x(P^{TI})} \sum_{j\in J} \sum_{t=1}^{T} c_{j,t}\, x_{j,t}$$

• *Linear ordering variables* $(x_{i,j})_{i,j\in J^<}$

If the schedules of the dominant set considered are entirely defined by their task sequence, they can be described by binary variables $(x_{i,j})_{i,j\in J^<}$ that indicate for each pair of tasks (i,j) whether i completes before j. These variables are called **linear ordering variables**, since they encode a linear ordering of tasks, that is a function from J to $[1..n]$, which assigns to each task its position in the sequence. In contrast with the previous variables, linear ordering variables often requires additionnal variables to provide a linear formulation of the problem. Indeed, having enough variables to encode a schedule is not sufficient, one needs variables that allow to express the objective function as a linear function, and to translate constraints by linear inequalities. Therefore, such variables result often in MIP formulations, and not IP formulations.

The fundamental non-overlapping constraints can be formulated using starting time variables as well as linear ordering variables by linear inequalities (7) and (8) presented below.

Even if Biskup and Feldmann propose a heuristic method for CDDP in [9], a MIP formulation for CDDP is also provided. Let us present below this formulation, that we solve in Section 2.3. For each task $j \in J$, three continuous variables are additionally considered: e_j representing its earliness, t_j representing its tardiness, and s_j representing its starting time. The following inequalities (3 – 6) ensure that these variables are consistent.

$$\forall j \in J,\ e_j \geqslant 0 \tag{3}$$

$$\forall j \in J,\ t_j \geqslant 0 \tag{4}$$

$$\forall j \in J,\ e_j \geqslant d - (s_j+p_j) \tag{5}$$

$$\forall j \in J,\ t_j \geqslant (s_j+p_j) - d \tag{6}$$

$$\forall (i,j) \in J^<,\ s_j+p_j \leqslant s_i + p(J)\ x_{i,j} \tag{7}$$

$$\forall (i,j) \in J^<,\ s_i+p_i \leqslant s_j + p(J)\ (1-x_{i,j}) \tag{8}$$

$$\forall (i,j) \in J^<,\ x_{i,j} \geqslant 0 \tag{9}$$

$$\forall (i,j) \in J^<,\ x_{i,j} \leqslant 1 \tag{10}$$

Let us introduce the polyhedron defined by these inequalities, along with the set of its integer points.

$$\boldsymbol{P^{LO}} = \left\{ (e,t,s,x) \in (\mathbb{R}^J)^3 \times \mathbb{R}^{J^<} \;\middle|\; (3-10) \text{ are satisfied} \right\} \quad \text{int}_{\boldsymbol{x}}(\boldsymbol{P^{LO}}) = \left\{ (e,t,s,x) \in P^{LO} \;\middle|\; x \in \{0,1\}^{J^<} \right\}$$

The linear ordering formulation for CDDP is then the following.

$$F_{\text{LO}} : \quad \min_{(e,t,s,x)\in(P^{LO})} \sum_{j\in J} \alpha_j \, e_j + \beta_j \, t_j$$

In this thesis, we use time-indexed and linear-ordering variables in formulations F_{LO} and F_{TI} for comparison. In contrast we use natural variables which are similar to completion times variables to provide formulation for common due dates problem in Part A. Moreover, we also use partition variables, which are usually used in other frameworks than scheduling field. The partition variables are introduced in a scheduling framework in the next section, in order to provide a compact MIP formulation for UCDDP.

0.5 A compact MIP formulation for the unrestrictive case

In this section, we focus on UCDDP. Thanks to the dominance properties, we show that UCDDP can be formulated as a partition problem, then we derive a compact linear MIP formulation. Finally, we exhibit a lack of dominance properties in the general case, which does not allow us to use the same approach for CDDP.

Let us consider a given instance UCDDP(p, α, β) in this section. Although we can only consider V-shaped d-blocks since they form a dominant set, in order to simplify the writing of the following formulations, we consider a larger dominant set: let $\boldsymbol{\mathcal{V}}$ be the set of V-shaped blocks with a task starting or completing at time d. Note that in such schedules, each tardy task is entirely processed after d, since there is no straddling task, *i.e.* there is no task starting before d and completing after d. Therefore, the early (resp. tardy) task set can be referred to the **left side** (resp. **right side**) of d.

0.5.1 Formulation of the UCDDP as a partition problem

In general, a schedule $\mathcal{S} \in \mathcal{V}$ cannot be encoded by its early-tardy partition. Indeed, if two early (resp. tardy) tasks have the same α-ratio (resp. β-ratio), one cannot determine which one is processed first, they can be sequenced arbitrarily. However, swapping two such tasks in $\mathcal{S}$ results in another schedule $\mathcal{S}' \in \mathcal{V}$ having the same total penalty. Furthermore, all the schedules in $\mathcal{V}$ having the same early-tardy partition have the same total penalty. Based on this remark, two schedules in $\mathcal{V}$ having the same early-tardy partition will be said **equivalent**. We denote by $\sim$ this relation.

We define an **ordered bi-partition** of a set A as a couple (A^1, A^2) where $\{A^1, A^2\}$ is a partition of A, *i.e.* $A^1 \cap A^2 = \emptyset$ and $A^1 \cup A^2 = J$. This is not only a partition into two subsets since the two subsets are not symmetric, *i.e.* $(A^1, A^2) \neq (A^2, A^1)$. Let $\vec{\mathcal{P}}_2(J)$ denote the set of the ordered bi-partitions of J. Note that the early-tardy partition of any schedule is an ordered bi-partition. Moreover, there is a one-to-one correspondence between equivalence classes for $\sim$ and ordered bi-partitions (E, T) of J, *i.e.* between $\mathcal{V}/\sim$ and $\vec{\mathcal{P}}_2(J)$. Therefore, we will say that a schedule $\mathcal{S} \in \mathcal{V}$ is a **representative** of an ordered bi-partition (E, T) if it belongs to the corresponding equivalence class, that is if its early-tardy partition is (E, T). In the sequel, we will only say **partition** to refer to an ordered bi-partition or to an early-tardy partition, when there is no ambiguity.

For any partition $(E, T) \in \vec{\mathcal{P}}_2(J)$, let $\boldsymbol{f}(E, T)$ denote the penalty of the equivalence class (E, T), that is the penalty of any representative of (E, T). From now on, our aim is to find a partition of J minimizing f, since the UCDDP can be formulated as follows.

$$F^1 : \min_{(E,T) \in \vec{\mathcal{P}}_2(J)} f(E, T)$$

We propose in the next section a linear translation of F^1. More precisely, we provide a compact MIP formulation for the UCDDP.

0.5.2 A compact MIP translating F^1

- *Providing an explicit expression of $f(E, T)$ based on a particular representative*

In order to provide an explicit expression of $f(E, T)$ for any partition (E, T), we will focus on a specific representative of (E, T), for which we can express task earliness and tardiness, and hence the total penalty.

For this purpose, we introduce two orders on J. Let us consider ρ and σ two bijective functions from

$[1..n]$ to J, such that $\left(\frac{\alpha_{\rho(k)}}{p_{\rho(k)}}\right)_{k\in[1..n]}$ and $\left(\frac{\beta_{\sigma(k)}}{p_{\sigma(k)}}\right)_{k\in[1..n]}$ are non-increasing.

A schedule is said **ρ-σ-shaped** if the early (resp. tardy) tasks are processed in decreasing order of ρ^{-1} (resp. increasing order of σ^{-1}).

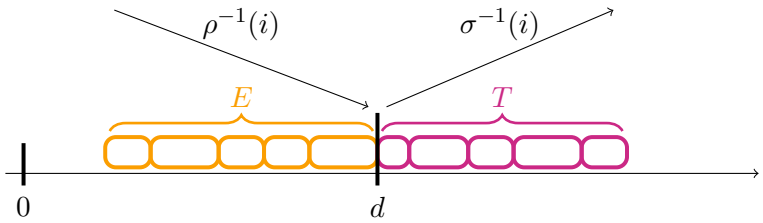


Figure 12: A ρ-σ-shaped d-block

Note that, $\rho(1)$ (resp. $\rho(n)$) is the index of the task having the largest (resp. the smallest) α-ratio: if this task is early in a ρ-σ-shaped d-block, it is necessarily the on-time task (resp. the first task). Similarly, $\sigma(1)$ (resp. $\sigma(n)$) is the index of the task having the largest (resp. the smallest) β-ratio: if this task is tardy in a ρ-σ-shaped d-block, it is necessarily the first (resp. the last) tardy task.

Note that a ρ-σ-shaped schedule is a V-shaped schedule. More precisely, each equivalence class of $\mathcal{V}$ admits a unique ρ-σ-shaped representative. Furthermore, the earliness and the tardiness of a task $j\in J$ in the representative of a partition (E,T) are given by the following expressions.

$$E_j = \sum_{\substack{i\in E\\ \rho^{-1}(i)<\rho^{-1}(j)}} p_i = \sum_{k=1}^{\rho^{-1}(j)-1} p_{\rho(k)}\,\mathbb{I}_{\rho(k)\in E} \quad \text{if } j\in E, \quad 0 \text{ otherwise}$$

$$T_j = p_j + \sum_{\substack{i\in T\\ \sigma^{-1}(i)<\sigma^{-1}(j)}} p_i = p_j + \sum_{k=1}^{\sigma^{-1}(j)-1} p_{\sigma(k)}\,\mathbb{I}_{\sigma(k)\in T} \quad \text{if } j\in T, \quad 0 \text{ otherwise}$$

Then, for any partition (E,T) we have the following explicit expression of $f(E,T)$.

$$f(E,T)=\sum_{j\in J}\alpha_j\,E_j+\beta_j\,T_j=\sum_{j\in J}\alpha_j\left(\mathbb{I}_{j\in E}\sum_{k=1}^{\rho^{-1}(j)-1}p_{\rho(k)}\,\mathbb{I}_{\rho(k)\in E}\right)+\beta_j\left(\mathbb{I}_{j\in T}\left(p_j+\sum_{k=1}^{\sigma^{-1}(j)-1}p_{\sigma(k)}\,\mathbb{I}_{\sigma(k)\in T}\right)\right)$$

- *Using binary variables to encode a partition*

In order to replace terms like $\mathbb{I}_{j\in E}$ or $\mathbb{I}_{j\in T}$ in the previous expression, let us introduce a vector δ of n binary variables encoding a partition. Given $\delta\in\{0,1\}^J$, we define the partition encoded by δ as $\big(E(\delta),T(\delta)\big)$, where $\boldsymbol{E(\delta)}=\{j\in J\mid\delta_j=1\}$ and $\boldsymbol{T(\delta)}=\{j\in J\mid\delta_j=0\}$. In other words, for each $j\in J$, δ_j indicates whether j is early or not, therefore $\mathbb{I}_{j\in E}=\delta_j$ and $\mathbb{I}_{j\in T}=(1-\delta_j)$. Hence, in a schedule $\mathcal{S}\in\mathcal{V}$ whose partition is encoded by δ, the earliness and the tardiness of a task $j\in J$ are given by the following expressions.

$$E_j=\delta_j\sum_{k=1}^{\rho^{-1}(j)-1}p_{\rho(k)}\,\delta_{\rho(k)} \quad\text{and}\quad T_j=(1-\delta_j)\left(p_j+\sum_{k=1}^{\sigma^{-1}(j)-1}p_{\sigma(k)}\,(1-\delta_{\sigma(k)})\right)$$

• *Linearization of the quadratic terms using X variables*

In order to replace the quadratic terms appearing in the previous earliness and tardiness expressions, let us introduce a vector of binary variables X indexed by $\boldsymbol{J^<} = \{(i,j) \in J^2 \mid i<j\}$. These variables can be used to linearize products of type $\delta_i\,\delta_j$, provided that they satisfy some linear constraints. The following lemma sums up this result established by Fortet [15].

Lemma 0.5 (*[15]*)

Let $(\delta, X) \in \mathbb{R}^J \times \mathbb{R}^{J^<}$.
If $\delta \in \{0,1\}^J$ *and* (δ, X) *satisfies the following inequalities:*

$$\forall (i,j) \in J^<,\ \ X_{i,j} \geqslant \delta_i - \delta_j \tag{X.1}$$

$$\forall (i,j) \in J^<,\ \ X_{i,j} \geqslant \delta_j - \delta_i \tag{X.2}$$

$$\forall (i,j) \in J^<,\ \ X_{i,j} \leqslant \delta_i + \delta_j \tag{X.3}$$

$$\forall (i,j) \in J^<,\ \ X_{i,j} \leqslant 2 - (\delta_i + \delta_j) \tag{X.4}$$

then $\forall (i,j) \in J^<, X_{i,j} = \begin{cases} 1 \text{ if } \delta_i \neq \delta_j \\ 0 \text{ otherwise} \end{cases}$, $\delta_i\,\delta_j = \dfrac{\delta_i + \delta_j - X_{i,j}}{2}$ *and* $(1\text{–}\delta_i)(1\text{–}\delta_j) = \dfrac{2 - (\delta_i + \delta_j) - X_{i,j}}{2}$.

The proof can be easily done by considering for each $(i,j) \in J^<$ two cases: $\delta_i = \delta_j$ and $\delta_i \neq \delta_j$. One can also notice that if (δ, X) satisfies (X.1-X.4), then $\delta \in [0,1]^J$ and $X \in [0,1]^{J^<}$.

Let us consider the polyhedron $\boldsymbol{P^n_{F^2}} = \left\{(\delta, X) \in \mathbb{R}^J \times \mathbb{R}^{J^<} \mid (X.1\text{-}X.4)\right\}$ and the set of its integer points $\mathrm{int}_{\boldsymbol{\delta}}(\boldsymbol{P^n_{F^2}}) = P^n_{F^2} \cap \{0,1\}^J \times \{0,1\}^{J^<}$. From Lemma 0.5, if a partition (E,T) is encoded by δ, then there exists a unique X such that $(\delta, X) \in \mathrm{int}_\delta(P^n_{F^2})$. We will say that (δ, X) encodes (E,T).

Using the Fortet linearization proposed in Lemma 0.5, the earliness and the tardiness of a task $j \in J$ in a ρ-σ-shaped d-block whose partition is encoded by (δ, X) are given by the following expressions[9].

$$E_j = \sum_{k=1}^{\rho^{-1}(j)-1} p_{\rho(k)}\,\frac{\delta_j + \delta_{\rho(k)} - X_{j,\rho(k)}}{2} \quad \text{and} \quad T_j = \sum_{k=1}^{\sigma^{-1}(j)-1} p_{\sigma(k)}\,\frac{2 - (\delta_j + \delta_{\sigma(k)}) - X_{j,\sigma(k)}}{2} + p_j(1-\delta_j)$$

Hence, the penalty of the partition encoded by $(\delta, X) \in \{0,1\}^J \times \{0,1\}^{J^<}$ is given by the following function $h_{\alpha,\beta}$, which is linear.

$$\boldsymbol{h_{\alpha,\beta}}(\delta, X) = \sum_{j \in J} \alpha_j \left(\sum_{k=1}^{\rho^{-1}(j)-1} p_{\rho(k)}\,\frac{\delta_j + \delta_{\rho(k)} - X_{j,\rho(k)}}{2} \right) + \beta_j \left(\sum_{k=1}^{\sigma^{-1}(j)-1} p_{\sigma(k)}\,\frac{2 - (\delta_j + \delta_{\sigma(k)}) - X_{j,\sigma(k)}}{2} + p_j(1-\delta_j) \right)$$

Finally, UCDDP can be formulated as the following compact MIP.

$$F^2 : \min_{(\delta,X) \in \mathrm{int}_\delta(P^n_{F^2})} h_{\alpha,\beta}(\delta, X)$$

Formulation F^2 is a direct linear translation of F^1. Indeed, there is a one to one correspondence between their solution sets, *i.e.* between $\mathrm{int}_\delta(P^n_{F^2})$ and $\vec{\mathcal{P}}_2(J)$, and $g(\delta, X) = f(E, T)$ for any (δ, X) encoding (E,T).
The advantage of F^2 is that it can be solved using a MIP solver such as CPLEX. Section 2.3 presents experimental results for different MIP formulations for UCDDP, including F^2. The drawback of F^2 is that it cannot be easily extended to CDDP. The following section explains why.

Note that polyhedron $P^n_{F^2}$ does not depend on ρ or σ. It is an extended polytope of the classical cut polytope for the complete undirected graph on the node set J [7], *Cf.* Chapter 3.

[9] For the sake of readability, we use $X_{i,j}$ regardless of whether $i<j$, that is to denote the variable $X_{\min(i,j),\max(i,j)}$.

0.5.3 Why F^2 cannot be extended to the general case

As explained in Section 0.5.2, we can compute the best d-block with respect to a given early-tardy partition. Conversely, computing the best left-block with respect to a given early-tardy partition is not straightforward, since we cannot say *a priori* which is the straddling task among the tardy tasks. Therefore, the early-tardy partition is no longer sufficient to encode an optimal schedule in the general case.

Let us consider the best left-block with respect to a given partition (E, T). Then the time between the starting time of the straddling task and d, denoted by a, is equal to $d - p(E)$ and the straddling task belongs to $\{ j \in T \mid p_j > a \}$.
One can conjecture that the straddling task maximizes β_j / p_j over this set. However, it is not the case, as shown by Counter-example 2 (*Cf.* Page 45).

Another conjecture can be done from Counter-example 2. Indeed, in this counter example, the non-optimality of the schedule where the first tardy task is the one with the largest β-ratio 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, in the optimal schedule (*i.e.* $\mathcal{S}^E$), the first tardy task (*i.e.* task 7) has the largest ratio among the tardy tasks longer than a (indeed, $\frac{\beta_7}{p_7 - a} = \frac{8}{1} > \frac{11}{2} = \frac{\beta_8}{p_8 - a}$). Then one can conjecture that the straddling task maximizes $\beta_j / (p_j - a)$ over tardy tasks with a processing time larger than a, *i.e.* $\{ j \in T \mid p_j > a \}$. Unfortunately, this is also false, as shown by Counter-example 3 (*Cf.* Page 46).

Counter-example 2 : *Schedules where the first tardy task is the one with the largest β-ratio are not dominant for* CDDP

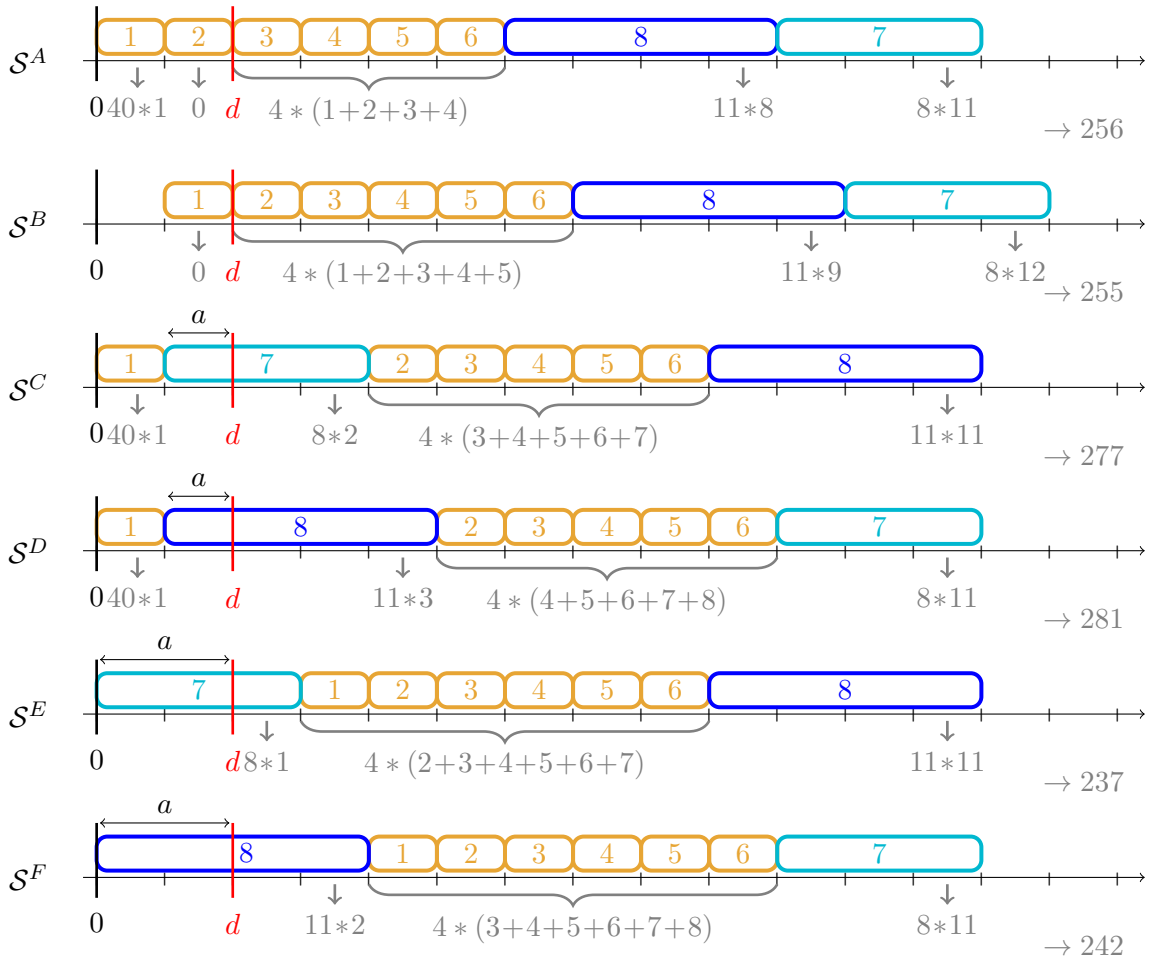
Let us consider the following instance of CDDP, where α_7 and α_8 can be chosen arbitrarily[10].

$$J=[1..8],\ d=2,\ \forall i\in[1..6],\ p_i=1,\ \alpha_i=40\ ,\ \beta_i=4\ ,\ \frac{\beta_i}{p_i}=4,$$
$$p_7=3,\ \alpha_7=1\ ,\ \beta_7=8\ ,\ \frac{\beta_7}{p_7}=\frac{8}{3},$$
$$p_8=4,\ \alpha_8=10^5,\ \beta_8=11,\ \frac{\beta_8}{p_8}=\frac{11}{4}$$

Note that this instance admits a lot of symmetries. Since tasks 1 to 6 are identical, interchanging them leads to a lot ($6!=720$ exactly) of equivalent schedules. We can say that such schedules are equivalent since they have the same total penalty, and even the same total earliness (resp. tardiness) penalty. In the sequel, we will choose to place them in the order given by their indices to reduce the number of schedules to consider. In other words, we will consider only one representative for each equivalence class.

According to Lemma 0.2, an optimal schedule can be found among the d-or-left-blocks. The following figure shows all the different types of d-or-left-blocks for this instance. Indeed, tasks 7 and 8 are too long to be early, then the set of early tasks only contains at most two *small* tasks, *i.e.* tasks belonging to [1..6].

- $\mathcal{S}^A$ represents the only type of V-shaped block where 2 small tasks are early.
- $\mathcal{S}^B$, $\mathcal{S}^C$, and $\mathcal{S}^D$ represent the three types of V-shaped d-or-left-block where 1 small task is early.
- $\mathcal{S}^E$, and $\mathcal{S}^F$ represent the two types of V-shaped d-or-left-block where no task is early.



In the optimal schedule type $\mathcal{S}^E$, the first tardy task is 7 while the task with the largest β-ratio in $\{\, j\in T \mid p_j > a \,\}$ is 8 since $a=2$ (7 starts at time $0 = d-2$), $p_8 > 2$ and $\frac{\beta_8}{p_8} = \frac{11}{4} > \frac{8}{3} = \frac{\beta_7}{p_7}$. Moreover, schedules where the first tardy task is task 8, that is schedules $\mathcal{S}^D$ and $\mathcal{S}^F$, are not optimal.

[10] since task 7 and 8 will never complete before $d=2$.

Counter-example 3 : *Schedules where the first tardy task is the one with the largest ratio* $\frac{\beta_j}{p_j - a}$ *are not dominant for* CDDP

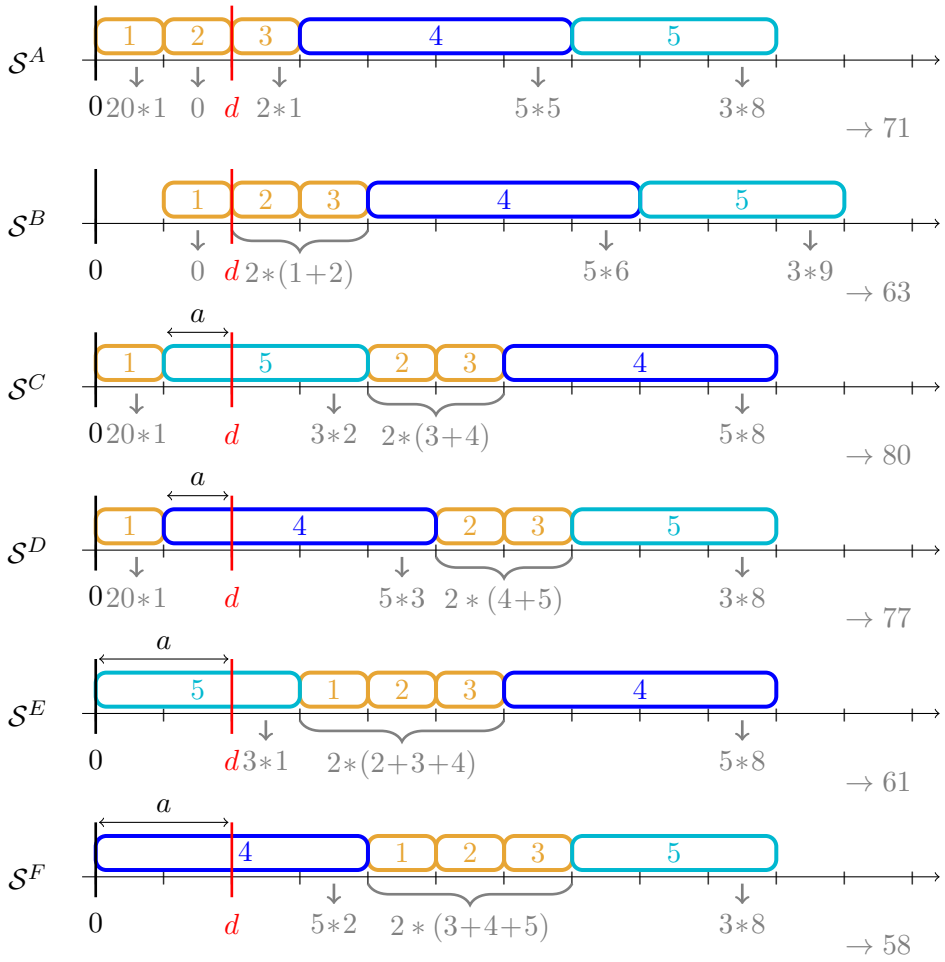
Let us consider the following instance of CDDP, where α_4 and α_5 can be chosen arbitrarily.

$$J = [1..5],\ d = 2,\ \forall i \in [1..3],\ p_i = 1,\ \alpha_i = 20\ ,\ \beta_i = 2$$
$$p_4 = 4,\ \alpha_4 = 1\ \ ,\ \beta_4 = 5$$
$$p_5 = 3,\ \alpha_5 = 10^5,\ \beta_5 = 3$$

As in the previous counter-example, the instance admits a lot of symmetries: tasks 1, 2, and 3 can be interchanged without changing the earliness and tardiness penalties Therefore, we choose to place them in the order given by their indices to reduce the number of schedules to consider.

According to Lemma 0.2, an optimal schedule can be found among the d-or-left-blocks. The following figure shows all the different types of d-or-left-blocks for this instance. Indeed, tasks 4 and 5 are too long to be early, then the set of early tasks only contains at most two *small* tasks, *i.e.* tasks belonging to [1..3],.

- $\mathcal{S}^A$ represents the only type of V-shaped block where 2 small tasks are early.
- $\mathcal{S}^B$, $\mathcal{S}^C$, and $\mathcal{S}^D$ represent the three types of V-shaped d-or-left-block where 1 small task is early.
- $\mathcal{S}^E$, and $\mathcal{S}^F$ represent the two types of V-shaped d-or-left-block where no task is early.



In the optimal schedule type $\mathcal{S}^F$, the first tardy task is 4 while the task with the largest ratio $\frac{\beta_j}{p_j - a}$ in $\{\, j \in T \mid p_j > a \,\}$ is 5 since $a = 2$ (4 starts at time $0 = d - 2$), $p_5 > 2$ and $\frac{\beta_5}{p_5 - a} = \frac{3}{1} > \frac{5}{2} = \frac{\beta_4}{p_4 - a}$.

Moreover, schedules where the first tardy task is the one with the largest ratio $\frac{\beta_j}{p_j - a}$ are $\mathcal{S}^E$ and $\mathcal{S}^D$ (since $a = 1$ and $\frac{\beta_4}{p_4 - a} = \frac{5}{3} > \frac{3}{2} = \frac{\beta_5}{p_5 - a}$), and none of them is optimal.

0.6 Outline of Parts A, B, and C

According to the previous section observations, it seems not possible to extend Formulation F^2 to CDDP. That is a motivation to investigate formulations based on other variables than partition variables.

In Part A, we focus on variables representing directly the earliness and tardiness of tasks that we call natural variables. Thanks to these variables, we provide two MIP formulations, one for UCDDP, and one for CDDP.

Since the variables (δ, X) used in F^2 also appear in these formulations, we study in Part B the associated polytope which is a kind of "cut polytope". The idea is to find facet defining inequalities able to strengthen our formulations.

Finally in Part C, we use dominance properties for UCDDP to derive linear inequalities removing locally non-optimal solutions, in order to improve Formulation F^2. We also try to extend this latter approach to other combinatorial problems.

PART A

Formulations using natural variables

For a minimization problem, a **regular criterion** is an objective function which is, for any task j, a non-increasing function of C_j. That means that 0 is an attractive point, in the sense that the closer from 0 is the task, the lower is the penalty of the schedule. In particular, left-tight schedules are dominant for such problems. Since completion times variables measure, for each task, its distance to 0, it seems *"natural"* to use them.

Queyranne [34] studies these variables for the single machine scheduling problem when minimizing $\sum \omega_j C_j$, which is a regular criterion. More precisely, he proposes to encode a schedule using only these *"natural variables"*, and provides linear inequalities ensuring that such variables encode a feasible schedule. While ensuring the non-negativity constraint using C_j is straightforward ($C_j \geqslant p_j$ suffices), the non-overlapping constraint requires an exponential number of inequalities —and an additional extremality constraint. In the sequel, we will call these inequalities non-overlapping Queyranne's inequalities.

In the just-in-time scheduling context, the criteria are no longer regular. In particular, in the common due date problems, 0 is not an attractive point, but d is. Indeed, the penalty induced by a task reduces when the task gets closer to d. Therefore, it is *"natural"* to change the reference point, and hence use variables measuring the distance to d, instead of the distance to 0. More precisely, since tasks can be placed before, as well as after d, we will use two family of *"natural variables"*: the earliness variables, denoted by e, for the distance on the left of d the tardiness variables, denoted by t, for the the distance to on the right of d. More generally, we will call **natural variables** any variables similar to completion times variables.

We show how the feasibility of a schedule encoded by (e, t) variables can be ensured using linear inequalities derived from the original non-overlapping Queyranne's inequalities. In this way, we provide a MIP formulation for UCDDP, and another one, similar, for CDDP.

Part A is divided into two chapters: Chapter 1 only gives theoretical results, namely properties of non-overlapping inequalities, and then the mathematical formulations, while Chapter 2 focuses on how to solve such formulations in practice.

Chapter 1

Non-overlapping inequalities

The first section of this chapter presents the non-overlapping Queyranne's inequalities and the related results provided in [34]. The second section, discusses the limitations of these results, and provides two lemmas to extend them, namely to use non-overlapping Queyranne's inequalities in combination with other inequalities. Sections 1.3 and 1.4 follow the same pattern: first we build the formulation step by step, then we show that it is a valid formulation for UCDDP (resp. CDDP). After these two sections, we try to retrieve the main ideas on which they are based, and give in Section 1.5 some insights of other problems where the same method could be used.

Let us consider in this chapter a given set of tasks J, and their processing times $p \in {\mathbb{R}_+^*}^J$. Moreover, to facilitate some proofs, we assume that $\forall j \in J,\ p_j \geqslant 1$. Even if all processing times have to be multiplied by $1/\min p_j$, we can make this assumption without loss of generality.

1.1 Non-overlapping Queyranne's inequalities for minimizing the weighted sum of completion times

As explained in Section 0.1, for a single-machine problem, a schedule must only satisfy two constraints to be feasible: non-negativity and non-overlapping constraints. Therefore, a vector[1] $y \in \mathbb{R}^J$ encodes a feasible schedule by its completion times if and only if it satisfies the two following constraints.

$$\text{non-negativity} \qquad \forall j \in J,\ y_j \geqslant p_j \tag{1.0}$$

$$\text{non-overlapping} \qquad \forall (i,j) \in J^<,\ y_j \geqslant y_i + p_j \text{ or } y_i \geqslant y_j + p_i \tag{1.1}$$

Completion time variables allow an easy way to express feasibility at the expense of the non-linearity of constraints (1.1). However, Queyranne [34] introduces linear inequalities using completion times to handle the non-overlapping for minimizing weighted sum of completion times, *i.e.* for the problem $1 \,|\, - \,|\, \sum \omega_j C_j$. We first recall notations and results proposed by Queyranne [34] before generalizing them to a larger framework in the next section. For $S \subseteq J$ and $y \in \mathbb{R}^J$, let us consider the following notations.

$$\boldsymbol{S}^< = \left\{(i,j) \in S^2 \,|\, i<j\right\} \qquad \boldsymbol{y(S)} = \sum_{i \in S} y_i \qquad \boldsymbol{p * y(S)} = \sum_{i \in S} p_i y_i \qquad \boldsymbol{g_p(S)} = \frac{1}{2}\left(\sum_{i \in S} p_i\right)^2 + \frac{1}{2}\sum_{i \in S} p_i^2$$

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

[1] Here we use a vector denoted y instead of C since these constraints will be used in the following for other variables, which will not represent completion times.

$$\forall S \subseteq J,\ g_p(S) = \sum_{(i,j)\in S^<} p_i\, p_j + \sum_{j\in S} p_j^2 \tag{1.2}$$

$$\forall S \subseteq J,\ \forall i \in J \backslash S,\ \ g_p\Big(S \sqcup \{i\}\Big) = g_p(S) + p_i\Big(p(S) + p_i\Big) \tag{1.3}$$

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

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

Let Q (resp. P^Q) denote the set of all vectors encoding a feasible schedule by its completion times, (resp. the polyhedron defined by inequalities (Q0)). Figure 1.1 illustrates Q and P^Q for a 2-task instance.

$$\boldsymbol{Q} = \Big\{ y \in \mathbb{R}^J \,\Big|\, y \text{ satisfies (1.0) and (1.1)} \Big\} \qquad \boldsymbol{P^Q} = \Big\{ y \in \mathbb{R}^J \,\Big|\, y \text{ satisfies (Q0)} \Big\}$$

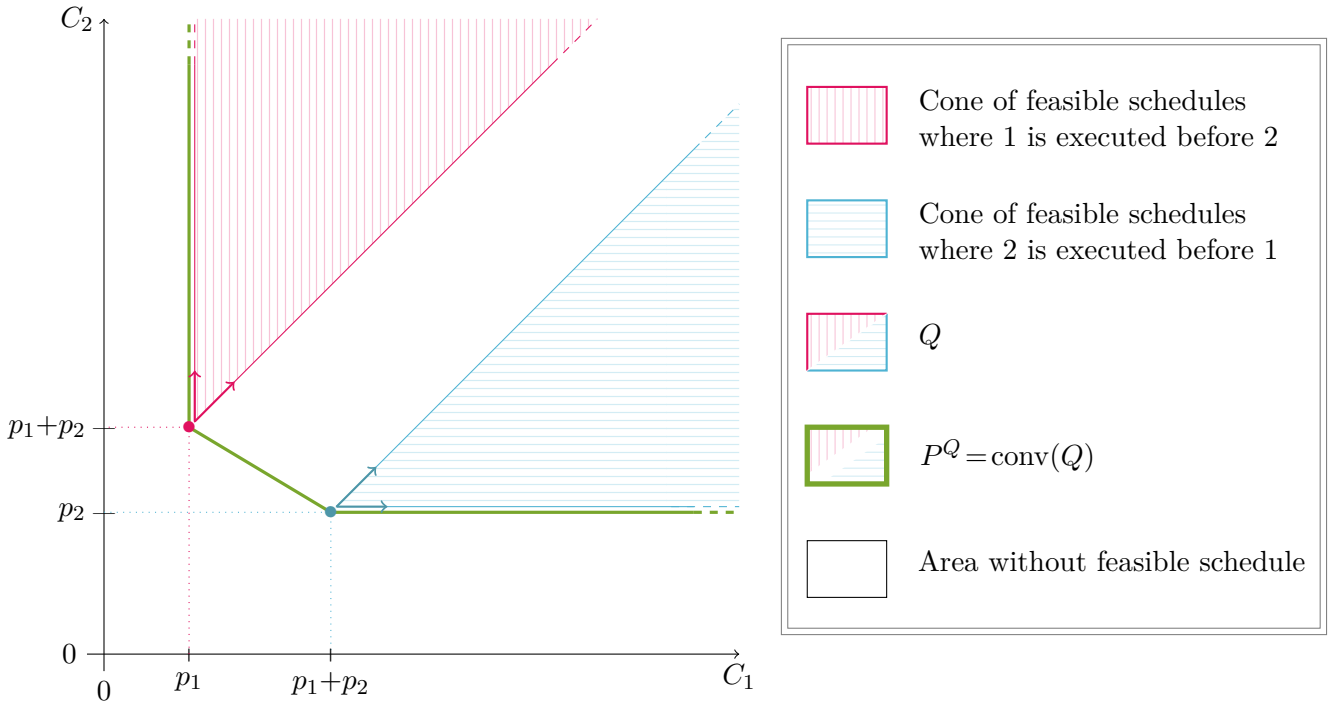


Figure 1.1: Q and conv(Q) in the case of two tasks

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

Property 1.1

Let $y \in \mathbb{R}^J$.
***If** y satisfies constraints (1.0) and (1.1), **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 $p_j > 0$. Hence, constraints (1.0) ensure that the inequalities (Q0) associated with the singletons are all satisfied.
- If $|S| \geqslant 2$, we need to exhibit an order on J. Since processing times are positive, constraints (1.1) ensure that $(y_j)_{j\in J}$ are distinct and then that there exists a (single) total order $\prec$ on J such that $i \prec j \Leftrightarrow y_i < y_j$. Then constraints (1.1) translate into $\forall (i,j) \in J^2,\ i \prec j \Rightarrow y_j \geqslant y_i + p_j$.

Using inequalities (1.0) 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 $|S| = k+1$. Let us set $j = \max_{\prec} S$ and $U = S \backslash \{j\}$. By induction, we have $p * y(U) \geqslant g_p(U)$, since $|U| = k-1$. Moreover, by previous arguments we have $y_j \geqslant p(U) + p_j$. 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 (1.3), hence y satisfies the inequality (Q0) associated with S.

□

As depicted in Figure 1.1, some points in $\mathrm{conv}(Q)$ correspond to unfeasible schedules Indeed, the two cones represent the set of feasible schedules, each corresponding to an order in the task execution, and points in between correspond to schedules where the two tasks overlap, then to unfeasible schedules. By definition of $\mathrm{conv}(Q)$, these points are in $\mathrm{conv}(Q)$, so they cannot be cut by the non-overlapping Queyranne's inequalities.
Moreover, one can observe that P^Q only has two extreme points and that they correspond to feasible schedules. This observation is true in general. Indeed, Queyranne [34] shows that the extreme points of P^Q always correspond to feasible schedules. The latter inclusion, *i.e.* $\mathrm{extr}(P^Q) \subseteq Q \subseteq \mathrm{conv}(Q)$, and the previous one, *i.e.* $\mathrm{conv}(Q) \subseteq P^Q$, are sufficient to say that $\min_{x \in Q} f(x) = \min_{x \in P^Q} f(x)$ for any linear function f, but not sufficient[2] to conclude that P^Q is exactly $\mathrm{conv}(Q)$. Queyranne [34] shows this equality using a geometrical argument: the equality of the recession cones of these two polyhedra. The following theorem sums up these results.

Theorem 1.2 (*Queyranne [34]*)

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

Moreover, Queyranne [34] shows that each extreme point of P^Q encodes a left-block. Conversely each left-block is encoded by an extreme point of P^Q since, according to the Smith rule [40], it is the only point in Q (and then in $\mathrm{conv}(Q) = P^Q$) minimizing $\omega * C(J)$ for $\omega \in \mathbb{R}^J_+$ such that the tasks are scheduled by strictly decreasing ratio ω_j / p_j.

The Theorem 1.2's proof given in [34] uses the geometrical structure of $\mathrm{conv}(Q)$*, which is the convex hull of a finite number of cones. As soon as schedules are subject to additional constraints like deadlines (i.e.* $C_j \leqslant \tilde{d}_j$*), the set of the vectors encoding feasible schedules is no longer an union of cones. Therefore, this proof cannot be extended. In order to use non-overlapping inequalities combined with other inequalities, we provide in the next section two lemmas which somehow generalize Theorem 1.2(i).*

[2]Such a reasoning would be valid for polytopes, but P^Q is unbounded.

1.2 Key lemmas to use non-overlapping inequalities in a larger setting

In this section, we provide two lemmas which will be the key for showing the validity of formulations using non-overlapping inequalities, such as formulations F^3 and F^4 that we propose in the next sections. The first key lemma gives a new proof of Theorem 1.2*(i)*. In this lemma, we explain how a vector of P^Q can be slightly disrupted in two opposite directions without leaving P^Q if it encodes a schedule presenting an overlap. Such a vector is then on the segment between two other points of P^Q, and hence it is not an extreme point of P^Q. Figure 1.2 illustrates the two ways of disrupting the overlapping tasks so that the corresponding vectors stay in P^Q.

Lemma 1.3 (*first key lemma*)

Let $y \in \mathbb{R}^J$ *satisfying 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* $y^{+-} = y + \frac{\varepsilon}{p_i}\mathbb{I}_i - \frac{\varepsilon}{p_j}\mathbb{I}_j$ *and* $y^{-+} = y - \frac{\varepsilon}{p_i}\mathbb{I}_i + \frac{\varepsilon}{p_j}\mathbb{I}_j$ *also satisfy* (Q0).

Figure 1.2: Illustration of the schedule's 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 with 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 $p * y^{+-}(S) = p * y(S) + p_i\frac{\varepsilon}{p_i} \geqslant p * y(S) \geqslant g_p(S)$, since $\varepsilon \geqslant 0$.

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, which respectively ensure that $m_1 > 0$ and $m_2 > 0$.

Claim

Let $y \in \mathbb{R}^J$ satisfying inequalities (Q0) and $(i,j) \in J^2$.
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: By contradiction, let us assume that there exists a subset $S \subseteq J$ 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 (1.3). 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 $y_j \leqslant p(S)(\star)$ since $p_j > 0$.

Then we have:
$$\begin{aligned} p{*}y\,(S{\sqcup}\{i\}) &= p{*}y\,(S) + p_i\,y_i && \text{by definition of } p*y \\ &= g_p(S) + p_i\,y_i && \text{by assumption} \\ &\leqslant g_p(S) + p_i\,y_j && \text{since } y_i \leqslant y_j \\ &\leqslant g_p(S) + p_i\,p(S) && \text{by } (\star) \\ &< g_p(S) + p_i\,[p(S) + p_i] && \text{since } p_i > 0 \\ &= g_p(S \sqcup \{i\}) && \textit{Cf.}\ (1.3) \\ &\leqslant p{*}y\,(S{\sqcup}\{i\}) && \text{by (Q0)} \end{aligned}$$
Therefore, $p{*}y\,(S{\sqcup}\{i\}) < p{*}y\,(S{\sqcup}\{i\})$. A contradiction. ■

Claim

Let $y \in \mathbb{R}^J$ satisfying inequalities (Q0) and $(i,j) \in J^2$.
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: By contradiction, let us assume that 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 $y_i \leqslant p(S) (\star)$.

Then we have:
$$\begin{aligned} p{*}y\,(S{\sqcup}\{j\}) &= p{*}y\,(S) + p_j\,y_j && \text{by definition of } p*y \\ &= g_p(S) + p_j\,y_j && \text{by assumption} \\ &< g_p(S) + p_j\,[y_i + p_j] && \text{since } y_j < y_i + p_j \\ &\leqslant g_p(S) + p_j\,[p(S) + p_j] && \text{by } (\star) \\ &= g_p(S \sqcup \{j\}) && \textit{Cf.}\ (1.3) \\ &\leqslant p{*}y\,(S{\sqcup}\{j\}) && \text{by (Q0)} \end{aligned}$$
Therefore, $p{*}y\,(S{\sqcup}\{j\}) > p{*}y\,(S{\sqcup}\{j\})$. A contradiction. ■

Thanks to these two claims, we have $m_1 > 0$, $m_2 > 0$ and then $\varepsilon = \min(m_1, m_2) > 0$.

Remark 1.4

Note that any ε between 0 and $\min(m_1, m_2)$ is suitable in this proof.
Therefore, ε can be chosen as small as wanted. □

To obtain an alternative proof of Theorem 1.2(i), Lemma 1.3 can be reformulated as follows.

If *C is a vector of P^Q which gives the task 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 with singletons ensure the non-negativity, 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 general 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 1.3 may not satisfy the additional inequalities. Let us consider for example, the following additional inequalities for a constant M such that $M \geqslant p(J)$.

$$\forall j \in J,\ y_j \leqslant M \tag{1.4}$$

Let Q^M (resp. P^{Q^M}) denote the set of all vectors encoding by its completion times a feasible schedule completing before time M (resp. the polyhedron defined by inequalities (Q0) and (1.4)). Figure 1.3 illustrates Q^M and P^{Q^M} for a 2-task instance.

$$\boldsymbol{Q^M} = \left\{ y \in \mathbb{R}^J \,\middle|\, y \text{ satisfies (1.0), (1.1) and (1.4)} \right\} \qquad \boldsymbol{P^{Q^M}} = \left\{ y \in \mathbb{R}^J \,\middle|\, y \text{ satisfies (Q0) and (1.4)} \right\}$$

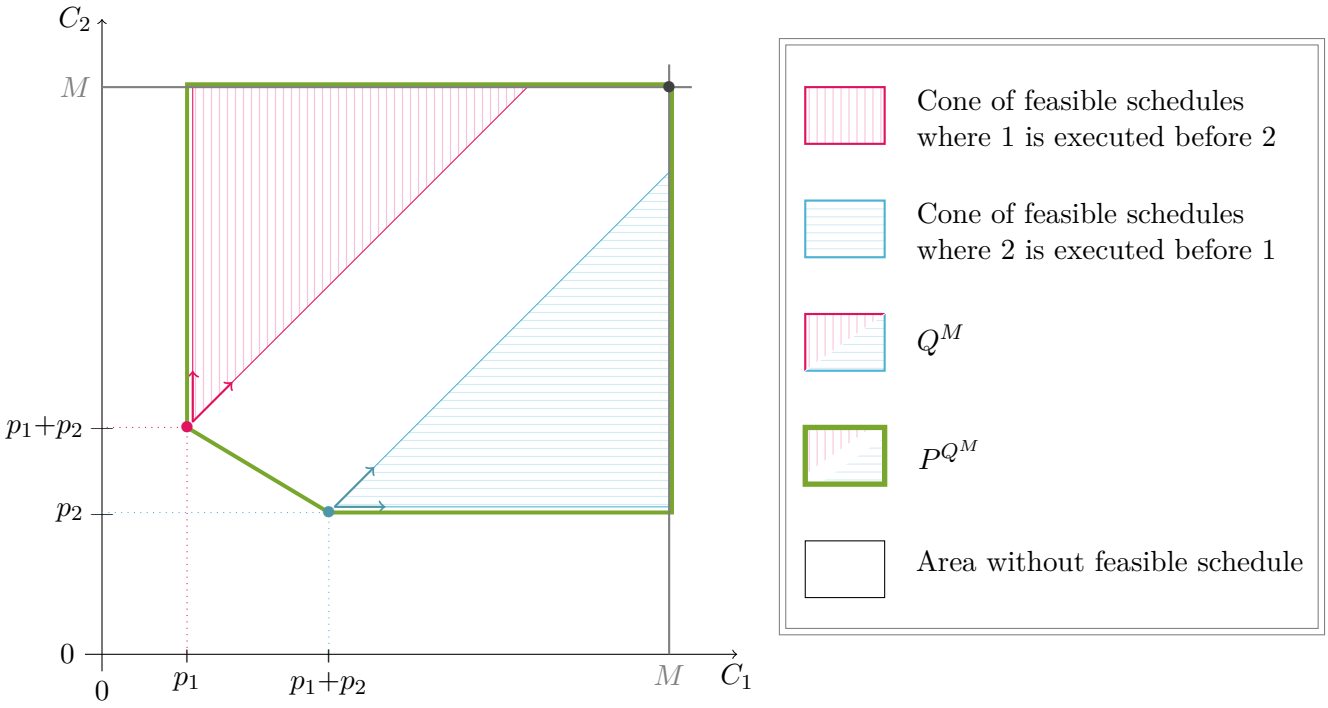


Figure 1.3: Q^M and P^{Q^M} in the case of two tasks

As depicted in Figure 1.3 for a 2-task instance, inequalities (1.4) induce extreme points encoding unfeasible schedules. Indeed, adding the inequalities $C_1 \leqslant M$ and $C_2 \leqslant M$ to the inequalities (Q0) leads to the extreme point (M, M), which encodes a schedule with an overlap.

Let us explain why applying Lemma 1.3 is not useful for this kind of extreme points in general. Let us assume that $C \in P^{Q^M}$ is the completion time vector of a schedule with two overlapping tasks. If one of these tasks completes at time M, applying Lemma 1.3 to C provides a vector C^{-+} which does not satisfy inequalities (1.4) and thus which is not in P^{Q^M}. Therefore, we cannot conclude this way that C is not an extreme point in P^{Q^M}.

However, provided that $\omega \in (\mathbb{R}_+^*)^2$, an extreme point like (M, M) is not an issue for the minimization of $\omega_1 C_1 + \omega_2 C_2$, since this point will not be proposed as an optimum. The idea is then the following: since the inclusion $\text{extr}(P^{Q^M}) \subseteq Q^M$ does not hold, we will prove that $\text{extr}^*(P^{Q^M}) \subseteq Q^M$ for a well-chosen subset of extreme points extr^*. More formally, we define extr^* for any given polyhedron $P \subseteq \mathbb{R}^n$ as follows.

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

In words, $\text{extr}^*(P)$ is defined as the set of points which are unique minimizer of a non-negatively weighted sum. The unicity is required to deal with some zero weight coefficients in the following. Since the extreme points of a polyhedron are exactly the points which can be written as the unique minimizer of a linear function, we have $\text{extr}^*(P) \subseteq \text{extr}(P)$.

The following lemma allows to handle the extreme points encoding a schedule with two overlapping tasks, when one of them completes at M. Indeed, this lemma will be the key for showing that such a point is not a unique minimizer.

Lemma 1.5 (*second key lemma*)

Let $y \in \mathbb{R}^J$ *satisfying inequalities* (Q0).
If *there exists* $(i,j) \in J^2$ *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* $y - \frac{\varepsilon}{p_j}\mathbb{I}_j$ *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 $y - \frac{\varepsilon}{p_j}\mathbb{I}_j$ 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 with 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.

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

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

Combining Lemmas 1.3 and 1.5, we prove that a point in $\mathrm{extr}^*\left(P^{Q^M}\right)$ encodes a feasible schedule. More precisely, we prove the following theorem.

Theorem 1.6

If $M \geqslant p(J)$, ***then*** $\mathrm{extr}^*\left(P^{Q^M}\right) \subseteq Q^M$.

Proof: Let $C \in \mathrm{extr}^*\left(P^{Q^M}\right)$. Since 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 1.3 for ε set in $]\,0, p(J) - C_j\,[$, so that C^{+-} and C^{-+} satisfy inequalities (Q0) and (1.4). 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 1.5, 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 non-negative 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 $\mathrm{extr}^*\left(P^{Q^M}\right)$. □

Despite this theorem is a kind of generalization of Theorem 1.2*(ii)*, it will not be used in the following. Conversely, Lemmas 1.3 and 1.5 will be.

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.*

1.3 A formulation for UCDDP using natural variables

In this section, we will provide a linear formulation for UCDDP. As explained in Section 0.2.1, we can consider an instance UCDDP(p, α, β) defined by $(p, \alpha, \beta) \in ({\mathbb{R}_+^*}^J)^3$ without loss of generality. Moreover, according to Lemma 0.1, we will only consider the d-blocks in this formulation.

> *NB*: All the inequalities useful for formulations F^3 and F^4 are summarized on page 217.
> I suggest to keep this page handy while reading the next two sections.

1.3.1 Building the formulation step by step

• *A linear objective function using e and t variables*

Since earliness and tardiness are not linear with respect to completion times, the function $f_{\alpha,\beta,d}$ 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. This way, the function $g_{\alpha,\beta}$ which gives the total penalty of a schedule from its earliness-tardiness vector (e, t) is linear. Indeed, this function is defined as follows.

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

In the remainder of this section, the objective function will be $g_{\alpha,\beta}$. To express the link between the latter and the previous objective function $f_{\alpha,\beta,d}$ for any given due date $d \in \mathbb{R}_+$, let us denote by θ_d the function which gives the earliness-tardiness vector of a schedule from its completion time vector C. Function θ_d is defined as follows.

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

Then the objective functions are linked as follows.

$$f_{\alpha,\beta,d} = g_{\alpha,\beta} \circ \theta_d$$

Note that the parameter d is omitted in the function $g_{\alpha,\beta}$. That is consistent with the omission of d in the definition of an UCDDP instance. Moreover, parameters α and β which are used here to define the optimization direction (*i.e.* the objective function), will not be used again in the rest of the section which is dedicated to describe the set of (e, t) vectors encoding d-blocks.

• *Consistency between e and t using δ variables*

We say that a vector (e, t) in $\mathbb{R}_+^J \times \mathbb{R}_+^J$ is **consistent** if $\forall j \in J$, either $e_j = 0$ or $t_j = 0$. For any $d \geqslant p(J)$, there exists C in $\mathbb{R}^J$ such that $\theta_d(C) = (e, t)$ if and only if (e, t) is consistent. In order to ensure consistency, we introduce the following inequalities using, as done for F_l^2, a variable δ_j indicating if j is early for each task $j \in J$. For each task j, δ_j indicates if j is early.

$$\forall j \in J,\ \ e_j \geqslant 0 \tag{1.5}$$
$$\forall j \in J,\ \ e_j \leqslant \delta_j\, (p(J) - p_j) \tag{1.6}$$
$$\forall j \in J,\ \ t_j \geqslant 0 \tag{1.7}$$
$$\forall j \in J,\ \ t_j \leqslant (1 - \delta_j)\, p(J) \tag{1.8}$$

Inequalities (1.5) and (1.6) 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 (1.6) does not restrict e_j when $\delta_j = 1$. Note that for an unrestrictive due date d, $p(J) - p_j$ is tighter than $d - p_j$. Similarly, inequalities (1.7) and (1.8) 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 1.7

Let $(e,t,\delta)\in\mathbb{R}^J\times\mathbb{R}^J\times\{0,1\}^J$.
If e,t,δ *satisfy inequalities (1.5 – 1.8),*
then (e,t) *is consistent and for any* $d\geqslant p(J)$, $C=\left(d-e_j+t_j\right)_{j\in J}$ *satisfies* $\theta_d(C)=(e,t)$.

Let θ_d^{-1} denote the function which gives the completion time vector of a schedule from its (necessarily consistent) earliness-tardiness vector (e,t).

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

In addition to the consistency, inequalities (1.5 – 1.8), ensure the non-negativity of the encoded schedule. Indeed, for any $d\in\mathbb{R}$ and $j\in J$, (1.6) and (1.7) ensure that $d-e_j+t_j\geqslant d-e_j\geqslant d-p(J)+p_j$. In particular, for any $d\geqslant p(J)$ we deduce that $d-e_j+t_j\geqslant p_j$. Hence, we obtain the following lemma.

Lemma 1.8

Let $(e,t,\delta)\in\mathbb{R}^J\times\mathbb{R}^J\times\{0,1\}^J$.
If e,t,δ *satisfy (1.5 – 1.8),* ***then*** *for any* $d\geqslant p(J)$, $\theta_d^{-1}(e,t)$ *satisfies* (1.0).

• *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. For a d-schedule, and even for any schedule with no straddling task, it suffices then that early (resp. tardy) tasks do not overlap each other. The global non-overlapping constraint is then decomposed into two non-overlapping constraints: one for each side of d.

In order to denote the early-tardy partition of a schedule from its completion times, we introduce the following notations.

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

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 non-negativity constraints for tardy tasks (resp. the non-overlapping constraint for tardy tasks). As shown in Figure 1.4, for an early task j, the value e_j+p_j can be seen as a completion time. Using $x_{/S}$ to denote $\left(x_j\right)_{j\in S}$ for any subset S of J and for any vector x in $\mathbb{R}^J$, the following lemma sums up these observations.

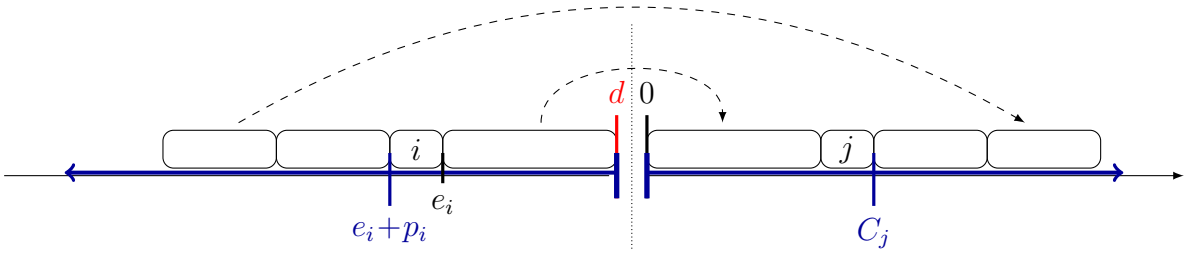


Figure 1.4: Illustration of the role of p_i+e_i for an early task i

Lemma 1.9

Let $d\in\mathbb{R}$ *such that* $d\geqslant p(J)$. *Let* $C\in\mathbb{R}^J$ *and* $(e,t)\in\mathbb{R}^J\times\mathbb{R}^J$.
If $(e,t)=\theta_d(C)$, ***then*** *the following equivalence holds.*

$$\left.\begin{array}{r} C \text{ satisfies (1.1)} \\ \forall j\in J,\ C_j\leqslant d \text{ or } C_j\geqslant d+p_j \end{array}\right\} \Leftrightarrow \begin{cases} (e+p)_{/E(C)} & \text{satisfies (1.0) and (1.1)} \\ t_{/T(C)} & \text{satisfies (1.0) and (1.1)} \end{cases}$$

Since the formulation do not use completion times, we use δ variables to obtain the early-tardy partition. Therefore, let us introduce the following notations to denote the early-tardy partition of a schedule from the δ variables.

$$\boldsymbol{E(\delta)} = \{\, j \in J \mid \delta_j = 1 \,\} \qquad\qquad \boldsymbol{T(\delta)} = \{\, j \in J \mid \delta_j = 0 \,\}$$

According to Section 1.1, we want to apply Queyranne's inequalities (Q0) to the vector $(e{+}p)_{/E(\delta)}$ (resp. $t_{/T(\delta)}$), so that it satisfies (1.0) and (1.1). Therefore, we consider the following inequalities.

$$\forall S \subseteq J, \qquad p * (e{+}p)\Big(S \cap E(\delta)\Big) \geqslant g_p\Big(S \cap E(\delta)\Big) \tag{1.9}$$

$$\forall S \subseteq J, \qquad p * t\Big(S \cap T(\delta)\Big) \geqslant g_p\Big(S \cap T(\delta)\Big) \tag{1.10}$$

These inequalities are not linear inequalities 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 (1.10) 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 (1.7) and (1.8). Since $p_i > 0$ and $p_j > 0$, this implies that $t_j > p_j$ and $t_j > p_i$, which is not valid, that is not true for some d-blocks.

To ensure that only the terms corresponding to early (resp. tardy) tasks are involved in the inequality (1.9) (resp. in (1.10)), we multiply each term of index j in S by δ_j (resp. by $(1{-}\delta_j)$). Then we obtain the following inequalities.

$$\forall S \subseteq J, \qquad \sum_{j \in S} p_j\, e_j\, \delta_j \geqslant \sum_{(i,j) \in S^<} p_i\, p_j\, \delta_i\, \delta_j \tag{1.11}$$

$$\forall S \subseteq J, \qquad \sum_{j \in S} p_j\, t_j\, (1{-}\delta_j) \geqslant \sum_{(i,j) \in S^<} p_i\, p_j\, (1{-}\delta_i)(1{-}\delta_j) + \sum_{j \in S} p_j^2\, (1{-}\delta_j) \tag{1.12}$$

For $\delta_j \in \{0, 1\}$, if (e, t, δ) satisfies inequalities (1.5 - 1.8), then we have $e_j\, \delta_j = e_j$ and $t_j\, (1{-}\delta_j) = t_j$. Therefore, the previous inequalities are simplified as follows.

$$\forall S \subseteq J, \qquad \sum_{j \in S} p_j\, e_j \geqslant \sum_{(i,j) \in S^<} p_i\, p_j\, \delta_i\, \delta_j \tag{1.13}$$

$$\forall S \subseteq J, \qquad \sum_{j \in S} p_j\, t_j \geqslant \sum_{(i,j) \in S^<} p_i\, p_j\, (1{-}\delta_i)(1{-}\delta_j) + \sum_{j \in S} p_j^2 (1{-}\delta_j) \tag{1.14}$$

• *Non-overlapping linear inequalities*

In order to remove the quadratic terms in the previous inequalities, we use the X variables introduced for F^2 in Section 0.5. Replacing $\delta_i\, \delta_j$ and $(1{-}\delta_i)(1{-}\delta_j)$ according to Lemma 0.5, 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 1.10

Let $(\delta, X) \in \{0,1\}^J \times \mathbb{R}^{J^<}$ satisfying inequalities (X.1 – X.4).

(i) ***If*** *$e \in \mathbb{R}^J$ satisfies inequalities (1.5) and (1.6) 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 (1.7) and (1.8) 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 1.9 and $(e+p)_{/E(\delta)}$ from Lemma 1.10 (resp. between $t_{/T(C)}$ and $t_{/T(\delta)}$).

Lemma 1.11

Let $(e,t,\delta)\in\mathbb{R}^J\times\mathbb{R}^J\times\{0,1\}^J$.
If *e,t,δ satisfy (1.5 - 1.8), and* (Q2)
then *for any* $d\geqslant p(J)$, $E(\delta)=\mathsf{E}\left(\theta_d^{-1}(e,t)\right)$ *and* $T(\delta)=\mathsf{T}\left(\theta_d^{-1}(e,t)\right)$.

Proof: Let $d\geqslant p(J)$ and $C=\theta_d^{-1}(e,t)$. Let $j\in J$.

– 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 (1.5), we deduce that $t_j>0$. Then inequality (1.8) implies that $\delta_j\neq 1$. Since $\delta_j\in\{0,1\}$, necessarily $\delta_j=0$. That proves $\mathsf{T}(C)\subseteq T(\delta)$.

– Conversely, if $j\in T(\delta)$, $\delta_j=0$ by definition. Inequalities (1.5) and (1.6) ensure then that $e_j=0$. 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 $T(\delta)\subseteq\mathsf{T}(C)$.

– If $j\in\mathsf{E}(C)$, then $C_j\leqslant d$ by definition. That is $t_j\leqslant e_j$ since $C_j=d-e_j+t_j$. From inequality (1.7), we deduce that $e_j\geqslant 0$. We distinguish then the two following cases.

- If $e_j=0$, then $t_j=0$ too from inequality (1.7). From inequality (Q2) for $S=\{j\}$, we deduce that $\delta_j\neq 0$, otherwise this inequality implies $t_j\geqslant p_j$, and hence $t_j>0$.
- If $e_j>0$, inequality (1.6) implies that $\delta_j\neq 0$, otherwise this inequality implies $e_j=0$.

In both cases, since $\delta_j\in\{0,1\}$, necessarily $\delta_j=1$. That proves $\mathsf{E}(C)\subseteq E(\delta)$.

– Conversely, if $j\in E(\delta)$, $\delta_j=1$ by definition. From inequality (1.8) we then have $t_j=0$, and hence $C_j=d-e_j$. Since $e_j\geqslant 0$ from inequality (1.5), we deduce that $C_j\geqslant d$. That proves $E(\delta)\subseteq\mathsf{E}(C)$. □

• *Formulation* F^3

Let us introduce the polyhedron defined by the linear inequalities introduced so far.

$$\boldsymbol{P^3}=\Big\{\ (e,t,\delta,X)\in\mathbb{R}^J\times\mathbb{R}^J\times[0,1]^J\times\mathbb{R}^{J^<}\ \Big|\ (1.5-1.8),\ (X.1-X.4),\ \text{(Q1) and (Q2) are satisfied}\ \Big\}$$

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 2.1.

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 $\mathrm{int}_{\boldsymbol{\delta}}$, which only keeps the integer points of any set $V\subseteq\mathbb{R}^J\times\mathbb{R}^J\times\mathbb{R}^J\times\mathbb{R}^{J^<}$.

$$\mathrm{int}_{\boldsymbol{\delta}}(\boldsymbol{V})=\Big\{(e,t,\delta,X)\in V\mid\delta\in\{0,1\}^J\Big\}.$$

As previously explained for P^Q and P^{Q^M}, some points in P^3 do not encode feasible schedules, even if they are integer (*i.e.* if $\delta\in\{0,1\}^J$). Therefore, we have to add an extremality constraint to our formulation, which is then not a classical MIP formulation. We finally propose the following formulation for UCDDP.

$$F^3:\quad\min_{(e,t,\delta,X)\in\mathrm{int}_\delta\left(\mathrm{extr}\,P^3\right)} g_{\alpha,\beta}(e,t)$$

The next section aims to prove the validity of F^3 *that is to show the two following statements.*

- *Each d-block is encoded by (at least) one point in* $int_\delta\left(\mathrm{extr}\,P^3\right)$*, and the value of this point according to* $g_{\alpha,\beta}$ *is the total penalty of the d-block.*
- *Each point in* $int_\delta\left(\mathrm{extr}\,P^3\right)$ *encodes a feasible d-block whose total penalty is given by* $g_{\alpha,\beta}$.

1.3.2 Validity of Formulation F^3

- *Why are there three theorems to show two statements?*

The main goal of this paragraph is to informally give the intuition of how the three following theorems are linked. To do this, let us use terms coming from the logic vocabulary. A deductive system is said **complete** if it allows to obtain all the true formulas, and it is said **sound** if it allows to obtain only true formulas. In some sense, to show the validity of our formulation, we want to show both its soundness and completeness.

- **Theorem 1.12** states that the (e, t, δ, X) encoding is complete, in the sense that any feasible schedule — under a technical assumption, which is satisfied in particular by any d-block — can be encoded by a point in $\text{int}_\delta(P^3)$.

- Conversely, a point in $\text{int}_\delta(P^3)$ does not necessarily encode a feasible schedule. However, if such point is also *(i)* an extreme point of (P^3) and *(ii)* a minimizer of $g_{\alpha,\beta}$, we can show that it encodes a feasible schedule, and even show that it is a d-block using Theorem 1.12. Therefore, **Theorem 1.13** states that formulation F^3 is sound, in the sense that any point in $\arg\min\{g_{\alpha,\beta} \,|\, \text{int}_\delta(\text{extr}(P^3))\}$ encodes a d-block.

- Then we need a stronger version of Theorem 1.12: we need to ensure that a feasible schedule is encoded by a vector in $\arg\min\{g_{\alpha,\beta} \,|\, \text{int}_\delta(\text{extr}(P^3))\}$, and not just in $\text{int}_\delta(P^3)$. This is not true for any feasible schedule. However, that is true for optimal d-blocks, *i.e.* C minimizing $f_{\alpha,\beta,d}(C)$. This is what **Theorem 1.14*(i)*** states.

- Finally, to conclude, we need a stronger version of Theorem 1.13: we need to ensure that a point in $\arg\min\{g_{\alpha,\beta} \,|\, \text{int}_\delta(\text{extr}(P^3))\}$ encodes an optimal d-block and not just a d-block. This can be shown using Theorem 1.14*(i)* (in the same way that we use Theorem 1.12 in the proof of Theorem 1.13 to show that the encoded schedule is a d-block rather than just a feasible schedule without straddling task). That is what **Theorem 1.14*(ii)*** states.

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

Theorem 1.12

Let $d \geqslant p(J)$. Let $C \in \mathbb{R}^J$.
***If** 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 $(e, t, \delta, X) \in int_\delta(P^3)$, such that $\theta_d(C) = (e, t)$, and hence $f_{\alpha,\beta,d}(C) = g_{\alpha,\beta}(e, t)$.*

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

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 the sake of brevity. Inequalities (1.5) and (1.7), as well as (1.6) for j in T and (1.8) 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 (1.6) for j in E (resp. inequalities (1.8) for j in T) are satisfied.
Using Lemma 0.5(i), X and δ satisfy inequalities $(X.1 - X.4)$.
Since C encodes a feasible schedule, C satisfies (1.1).Using Lemma 1.9, $(e+p)_{/E}$ (resp. $t_{/T}$) satisfies (1.0) and (1.1). Applying Property 1.1 to these two vectors, we deduce that they satisfy (Q0), and using Lemma 1.10, that e, δ, X satisfy (Q1) and t, δ, X satisfy (Q2). Thus, Y belongs to P^3, and even to $\text{int}_\delta(P^3)$ since $\delta \in \{0, 1\}^J$. □

The following theorem establishes that an optimal solution of formulation F^3 is a solution for the unrestrictive common due date problem. Let us define $\pi_{e,t}$ the projection of the whole variable space to the (e,t) variable space as follows.

$$\boldsymbol{\pi_{e,t}} = \begin{pmatrix} \mathbb{R}^J \times \mathbb{R}^J \times \mathbb{R}^J \times \mathbb{R}^{J^<} & \longrightarrow & \mathbb{R}^J \times \mathbb{R}^J \\ (e,t,\delta,X) & \longmapsto & (e,t) \end{pmatrix}$$

Theorem 1.13

Let $d \geqslant p(J)$. Let $Y^* = (e,t,\delta,X) \in int_\delta(P^3)$.
***If** $Y^* \in \text{extr}(P^3)$ and (e,t) minimizes $g_{\alpha,\beta}$ on $\pi_{e,t}\Big(int_\delta(P^3)\Big)$,*
***then** Y^* encodes a d-block whose total penalty is $g_{\alpha,\beta}(e,t)$.*

Proof: We decompose the proof into two steps.

– Proving that Y^* encodes a feasible schedule without straddling task
From Lemma 1.7, (e,t) is consistent and we can set $C^* = \theta_d^{-1}(e,t)$. Then Y^* 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 (1.0) and (1.1). From Lemma 1.8, C^* satisfies (1.0). From Lemma 1.11, $E(\delta) = \mathsf{E}(C^*)$ (resp. $T(\delta) = \mathsf{T}(C^*)$), which allows the notation E (resp. T) for the sake of brevity. Since $(e,t) = \theta_d(C^*)$, we can use Lemma 1.9: to show that C^* satisfies (1.1), and at the same time that there is no straddling task in $\mathcal{S}^*$, it remains to show that $(e+p)_{/E}$ (resp. $t_{/T}$) satisfies (1.0) and (1.1).

From Lemma 1.10, we know that $(e+p)_{/E}$ (resp. $t_{/T}$) satisfies inequalities (Q0). First, using in particular inequalities (Q0) associated with singletons, that ensures that $(e+p)_{/E}$ (resp. $t_{/T}$) satisfies (1.0). Second, that allow to show that $(e+p)_{/E}$ (resp. $t_{/T}$) satisfies (1.1) in the same way that we have shown, in Section 1.1, that a vector in $\text{extr}^*(P^{Q^M})$ encodes a schedule without overlapping. Let us assume that $(e+p)_{/E}$ does not satisfy (1.1). Then there exists $(i,j) \in E^2$ such that $e_i + p_i \leqslant e_j + p_j < (e_i + p_i) + p_j$. Depending on whether there is a gap between $e_j + p_j$ and the bound $p(J)$, we show that this overlap contradicts either the extremality of Y^* or the minimality of (e,t). Therefore, two cases have to be considered.

→ If $e_j + p_j < p(J)$, then from Lemma 1.3 applied on $(e+p)_{/E}$ there exists $\varepsilon \in \mathbb{R}_+^*$ such that setting

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

leads to two vectors $(e^{+-}+p)_{/E}$ and $(e^{-+}+p)_{/E}$ satisfying (Q0). According to Remark 1.4, ε can be chosen such that $\varepsilon \leqslant p(J) - (p_j + e_j)$, and hence such that $\varepsilon \leqslant p(J) - (p_i + e_i)$ $(\star)$.

Using Lemma 1.10, both e^{+-}, δ, X and e^{-+}, δ, X satisfy (Q1).

Since $p_i \geqslant 1$, we have $e_i^{+-} = e_i + \frac{\varepsilon}{p_i} \leqslant e_i + \varepsilon$, and using $(\star)$ we obtain $e_i^{+-} \leqslant p(J) - p_i$. For $k \in J \backslash \{i\}$, we have $e_k^{+-} \leqslant e_k$ and since e satisfies (1.6), we deduce that $e_k^{+-} \leqslant p(J) - p_k$. Thus e^{+-} satisfies (1.6).

For any $k \in E$, we have $e_k^{+-} + p_k \geqslant p_k$ since the inequality (Q0) associated with the singleton k is satisfied by $(e^{+-}+p)_{/E}$. For any $k \in T$, we have $e_k^{+-} = e_k$, and $e_k \geqslant 0$ since e satisfies inequalities (1.5). We deduce that e^{+-} satisfies inequalities (1.5).

Similarly, e^{-+} satisfies inequalities (1.5) and (1.6). Finally, the two following points are in P^3.

$$Y^{+-} = (e^{+-}, t, \delta, X) \quad \text{and} \quad Y^{-+} = (e^{-+}, t, \delta, X)$$

Then Y^* is the middle of two points of P^3. **A contradiction, since Y^* is extreme.**

→ If $e_j + p_j \geqslant p(J)$, then $e_j + p_j \geqslant p(E)$, and from Lemma 1.5 on $(e+p)_{/E}$ there exists $\varepsilon \in \mathbb{R}_+^*$ such that setting

$$e^{-} = e - \frac{\varepsilon}{p_j}\mathbb{I}_j$$

leads to a vector $(e^{-}+p)_{/E}$ satisfying (Q0).Using Lemma 1.10, e^{-}, δ, X satisfy (Q1).

Since e^{-} is component-wise smaller than e, e^{-} satisfies inequalities (1.6) as well.

For any $k \in E$, we have $e^-_k + p_k \geqslant p_k$ since the inequality (Q0) associated with the singleton k is satisfied by $(e^- + p)_{/E}$. For any $k \in T$, we have $e^-_k = e_k$, and $e_k \geqslant 0$ since e satisfies inequalities (1.5). We deduce that e^- satisfies inequalities (1.5).

Finally, the following point is in P^3.

$$Y^- = (e^-, t, \delta, X)$$

Since α and β parameters are positive, Y^- has a smaller value than Y^* according to $g_{\alpha,\beta}$, *i.e.* $g_{\alpha,\beta}(e^-, t) < g_{\alpha,\beta}(e, t)$. **A contradiction, since** (e, t) **minimizes** $g_{\alpha,\beta}$ **on** $\pi_{e,t}(P^3)$.

Since both cases leads to a contradiction, we deduce that $(e+p)_{/E}$ satisfies (1.1). In the same way, we can prove that $t_{/T}$ satisfies (1.1). As explained above, that suffices to show that $\mathcal{S}^*$ is a feasible schedule.

– Proving 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 it is a d-block.

Let us assume that $\mathcal{S}^*$ holds an idle time or has no early task. Let $\widehat{\mathcal{S}}$ denote the schedule obtained by tightening tasks around d so as to fill idle times between tasks and, if there is no early task, left-shifting all the tasks such that the first one becomes on-time. Since $d \geqslant p(J)$, this left-shifting operation respects the non-negativity constraint. Let $\widehat{C}$ denote the completion times in $\widehat{\mathcal{S}}$.

Since $\widehat{\mathcal{S}}$ is a d-block by construction, we have $\forall j \in J$, $d - p(J) \leqslant \widehat{C}_j - p_j$ and $\widehat{C}_j \leqslant d + p(J)$, which allows to use Theorem 1.12. We deduce that there exists $\widehat{Y} = (\hat{e}, \hat{t}, \hat{\delta}, \hat{X}) \in \text{int}_\delta(P^3)$, such that $\theta_d(\widehat{C}) = (\hat{e}, \hat{t})$.

Moreover, $\widehat{\mathcal{S}}$ has a lower penalty than $\mathcal{S}^*$, *i.e.* $f_{\alpha,\beta,d}(\widehat{C}) < f_{\alpha,\beta,d}(C^*)$, 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 $g_{\alpha,\beta}(\hat{e}, \hat{t}) = f_{\alpha,\beta,d}(\widehat{C}) < f_{\alpha,\beta,d}(C^*) = g_{\alpha,\beta}(e, t)$. A contradiction since (e, t) minimizes $g_{\alpha,\beta}$ on $\pi_{e,t}(\text{int}_\delta(P^3))$. □

The following theorem establishes that UCDDP reduces to solving formulation F^3.

Theorem 1.14

Let $d \geqslant p(J)$.
(i) Any optimal d-block is encoded by a vector minimizing $g_{\alpha,\beta}$ on $\text{int}_\delta\Big(\text{extr}\, P^3\Big)$.
(ii) Conversely, any vector minimizing $g_{\alpha,\beta}$ on $\text{int}_\delta\Big(\text{extr}\, P^3\Big)$ encodes an optimal d-block.

Proof: Let us consider an optimal d-block $\mathcal{S}^*$. From Theorem 1.12, there exists a vector $Y^* = (e^*, t^*, \delta^*, X^*)$ in $\text{int}_\delta(P^3)$ encoding $\mathcal{S}^*$. We will prove that Y^* is necessary an extreme point of P^3. We introduce $P^{\delta^*} = \{(e, t) \,|\, (e, t, \delta^*, X^*) \in P^3\}$, which is the slice of P^3 according to δ^*, *i.e.* the projection of set of points of P^3 satisfying $\delta = \delta^*$ and therefore $X = X^*$.

Claim

If (e^*, t^*) is an extreme point of P^{δ^*}, **then** Y^* is an extreme point of P^3.

Proof: Let us show the contrapositive implication. Hence, we assume that Y is not an extreme point in P^3. By definition, there exist then $Y^1 = (e^1, t^1, \delta^1, X^1)$ and $Y^2 = (e^2, t^2, \delta^2, X^2)$ in P^3 such that $Y^* = \frac{1}{2}(Y^1 + Y^2)$. Then δ^1 and δ^2 are necessarily equal to δ^* since $\delta^* \in \{0, 1\}^J$, $\delta^1 \in [0, 1]^J$, and $\delta^2 \in [0, 1]^J$. By Lemma 0.5, we deduce that $X^1 = X^*$ (resp. $X^2 = X^*$), and thus that (e^1, t^1) (resp. (e^2, t^2)) is in P^{δ^*}. Since $(e^*, t^*) = \frac{1}{2}((e^1, t^1) + (e^2, t^2))$, that shows that (e^*, t^*) is not 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 1.10, and inequalities (1.5 - 1.8) we decompose P^{δ^*} as the Cartesian product $P^{\delta^*} = P^{\delta^*, E} \times \{0\}^T \times P^{\delta^*, T} \times \{0\}^E$ where

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

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 1.1, 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 $M = p(J) \geqslant p(E)$.

Queyranne has shown that left-tight schedules are encoded by extreme points of P^Q. These points are also extreme points of P^{Q^M} since no task is scheduled after $M \geqslant p(J)$ in a left-tight schedule. Therefore, 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 Y^* encodes a d-block. We deduce that (e^*, t^*) belongs to $\text{extr}(P^{\delta^*})$ and thus that Y^* belongs to $\text{int}_\delta(\text{extr}\, P^3)$.

To prove item *(i)*, it remains to show that Y^*, or more precisely (e^*, t^*), is a minimizer of $g_{\alpha,\beta}$. By contradiction, let us assume that there exists $\widehat{Y} = (\hat{e}, \hat{t}, \hat{\delta}, \hat{Y}) \in \text{int}_\delta(\text{extr}\, P^3)$ 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 1.13, $\widehat{Y}$ 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 a direct corollary of item *(i)* and Theorem 1.13. The schedule encoded by a vector Y^* minimizing $g_{\alpha,\beta}$ on $\pi_{e,t}\left(\text{int}_\delta(\text{extr}\, P^3)\right)$ is a d-block, and if it is not optimal, there would exist a strictly better d-block, and a vector in $\text{int}_\delta(\text{extr}\, P^3)$ with a smaller value according to $g_{\alpha,\beta}$, a contradiction. □

The next section provides a formulation similar to F^3 for CDDP. The main ideas are the same, but the occurrence of a straddling task in dominant schedules makes the encoding and the validity proof a bit more difficult.

1.4 A formulation for CDDP using natural variables

As explained in Section 0.2.2, we can consider an instance CDDP(p, α, β, d) where $\beta \in (\mathbb{R}_+^*)^J$. Moreover, only d-or-left-blocks will be consider since they form a dominant set.

1.4.1 An encoding based on a new reference point

• *Encoding a schedule with a straddling task*

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 the one associated with $\{j\}$, *i.e.* $t_{j_s} \geqslant p_{j_s}$. 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 1.5 illustrates this encoding for a schedule holding a straddling task.

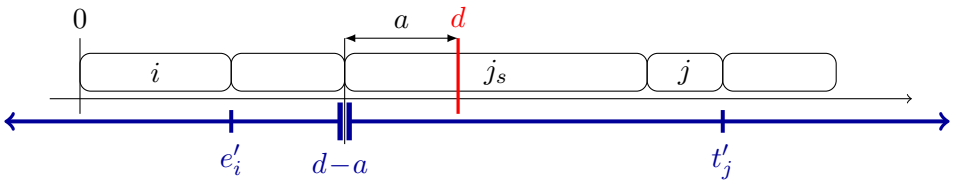


Figure 1.5: The (a, e', t') encoding for a schedule holding a straddling task j_s

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 schedules having an on-time task with variables a, e', t'.

• *Two possible encodings for a schedule with an on-time task*

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 F^3. Hence, a first possible encoding consists in setting $a = 0$, and using e' (resp. t') to represent earliness (resp. tardiness). Figure 1.6 illustrates this encoding for a schedule holding an on-time task.

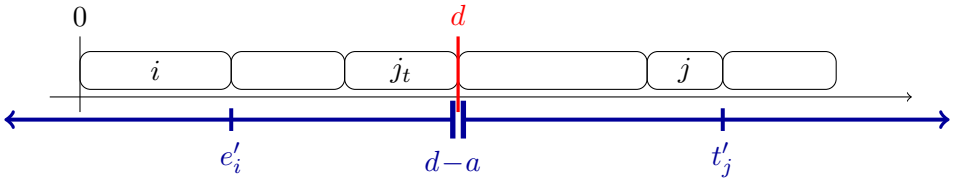


Figure 1.6: The first (a, e', t') encoding for a schedule holding an on-time task j_t

Unfortunately, to ensure that a takes the expected value in case of a schedule holding a straddling task, we will introduce a boolean variable to identify the task j_0 beginning at $d-a$. It forces to have in every schedule a task beginning at $d-a$. Therefore, this encoding proposed above is not valid in case of a d-block without tardy task, since no task starts at $d-a = d$ in such a schedule. We then propose a second encoding for the d-blocks. It consists in choosing the starting time of the on-time task 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 1.7.

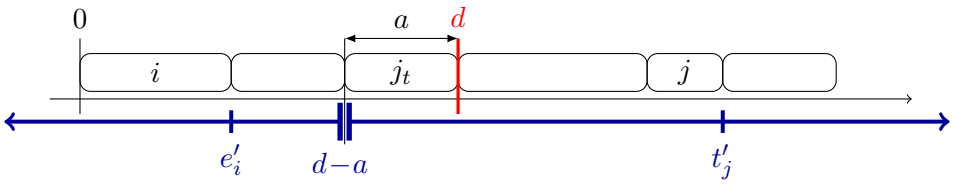


Figure 1.7: The second (a, e', t') encoding for a schedule holding an on-time task j_t

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, all the encoding proposed in this section can be decoded in the same way: the following expression gives the completion times of the encoded schedule.

$$C = \left(d - a - e'_j + t'_j\right)_{j \in J}$$

- *A task partition slightly different from the early-tardy partition*

Let us introduce the following notations in order to partition the tasks according to their completion times in a slightly different way than $(E(C), T(C))$ does.

$$\widetilde{E}(C) = \{\, j \in J \mid C_j < d \}\qquad\qquad \widetilde{T}(C) = \{\, j \in J \mid C_j \geqslant d \,\}$$

Figure 1.8 (resp. Figure 1.9) illustrates this partition for a schedule with a straddling task (resp. with an on-time task). Note that if there is a straddling task in the schedule, then $E(C) = \widetilde{E}(C)$ and $T(C) = \widetilde{T}(C)$.

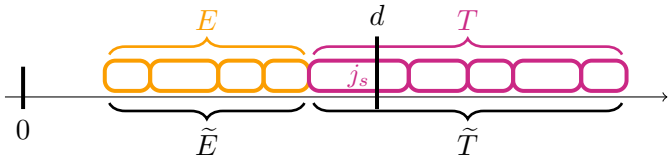


Figure 1.8: $(\widetilde{E}, \widetilde{T})$ when a straddling task occurs

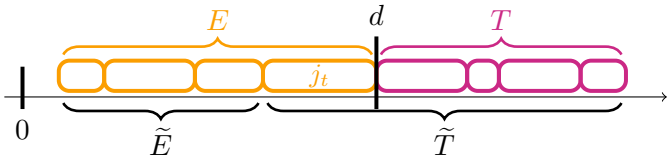


Figure 1.9: $(\widetilde{E}, \widetilde{T})$ when an on-time task occurs

1.4.2 Building the formulation step by step

- *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 (1.5 – 1.8) (*Cf.* page 58) where e (resp. t) are replaced by e' (resp. t'). These inequalities will be denoted by (1.5' – 1.8') in the sequel.

Note that δ_j no longer necessarily indicates if task j is early or not. In particular, $\left(E(\delta), T(\delta)\right)$ is no longer necessarily the early-tardy partition.

→ When a straddling task occurs, both partitions coincide, then we have:

$$E(\delta) = E(C) = \widetilde{E}(C) \quad \text{and} \quad T(\delta) = T(C) = \widetilde{T}(C)$$

→ When an on-time task occurs, using the first encoding we have

$$E(\delta) = E(C) \quad \text{and} \quad T(\delta) = T(C),$$

but using the second one we have

$$E(\delta) = \widetilde{E}(C) \quad \text{and} \quad T(\delta) = \widetilde{T}(C).$$

• *Handling the non-negativity*

Since the due date can be smaller than $p(J)$, avoiding overlaps and idle times does not ensure the non-negativity constraint. Therefore, we add the following inequalities ensuring that $e'_j + p_j \leqslant d-a$ for each task j completing before $d-a$.

$$\forall j \in J, \qquad e'_j + p_j\,\delta_j \leqslant d-a \tag{1.15}$$

Since d is an upper bound of a, *i.e.* $d-a \geqslant 0$, these inequalities are valid even for a task j completing after $d-a$, *i.e.* such that $\delta_j = 0$, and thus $e'_j = 0$.

• *Handling the non-overlapping*

To ensure the non-overlapping, we use again variables X, satisfying (X.1-X.4) and the inequalities (Q1) and (Q2) (*Cf.* page 60), 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 (1.15) 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 $M \geqslant p(J)$[3]. 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, as the following example shows.

Example 4 : *A constraint $C_j \leqslant M$ for a too small M disables the non-overlapping inequalities*

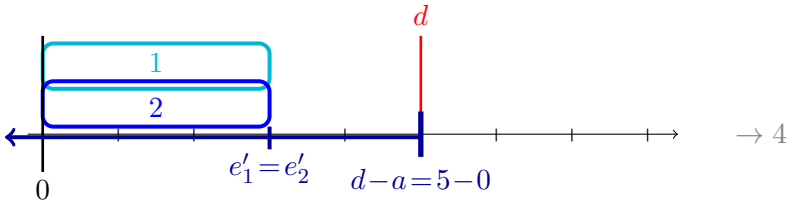
Let us consider the following instance of CDDP reduced to two identical tasks.

$$J = [1..2],\ \ d = 5,\ \ p_1 = 3,\ \ \alpha_1 = 1,\ \ \beta_1 = 10,$$
$$p_2 = 3,\ \ \alpha_2 = 1,\ \ \beta_2 = 10,$$

Let us consider the polyhedron corresponding to the draft formulation built so far.

$$P = \Big\{ (e', t', \delta, X, a) \in \mathbb{R}^J \times \mathbb{R}^J \times [0,1]^J \times \mathbb{R}^{J^<} \times \mathbb{R} \;\Big|\; (1.5'-1.8'),\ (X.1-X.4),\ (\text{Q1'}),\ (\text{Q2'}),\ \text{and}\ \ (1.15) \Big\}$$

The vector $Y = (2, 2, 0, 0, 1, 1, 0, 0)$, is an integer extreme point of P. It corresponds to the schedule $\mathcal{S}$ represented below, where both tasks complete at time 3, since $e'_1 = e'_2 = 2$ and $a = 0$.



The positivity constraint is satisfied, but not the non-overlapping constraint. Then $\mathcal{S}$ is unfeasible, then its penalty is not defined. However, one can say that $\mathcal{S}$ has a total penalty of 4, since $\alpha_1(e'1+a)+\alpha_2(e'2+a) = 4$. More precisely $\mathcal{S}$ minimizes the following function supposed to represent the total penalty over P.

$$\sum_{j \in J} \alpha_j\,(e' - j + a\delta_j) + \beta_j\,(e' - j - a(1-\delta_j))$$

Finally, Y encodes a schedule where an overlap occurs in spite of inequalities (Q1') and despite Y is both extreme an minimal. The reason is that $d-a = 5 < 6 = p(E(\delta))$, then Lemma 1.5 fails to be applied to $(e'+p)_{/E(\delta)}$. One can say that "*$d-a$ is a too restrictive big M*".

[3]Implicitely, inequalities (Q0) are written for the whole set of tasks J. In case where the non-overlapping only involves a subset of tasks $I \subseteq J$, the condition is $M \geqslant p(I)$.

To prevent $d-a$ to be too restrictive, we introduce the following inequality.

$$\sum_{j\in J} p_j\,\delta_j \leqslant d-a \tag{1.16}$$

Indeed, for a given $\delta\in\{0,1\}^J$, inequalities (1.16) for $j\in E(\delta)$ are similar to inequalities (1.4), where $d-a$ can play the role of M since $d-a\geqslant p\Big(E(\delta)\Big)$.

To ensure that inequalities (Q1') (resp. (Q2')) prevent overlaps of tasks completing before (resp. after) $d-a$, the objective function must be:

(i) a non-increasing function of variable e'_j for each task j such that $\delta_j=1$,

(ii) a non-increasing function of variable t'_j for each task j such that $\delta_j=0$.

Note that if a takes a value 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. The new point is dedicated to provide linear inequalities ensuring that the variable a takes a value such that the objective function fulfills *(i)* and *(ii)*.

• *Ensuring that variable a takes the expected value*

In spite of their apparent symmetry, the two conditions (i) and (ii) are completely different.

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

$$a\geqslant 0 \tag{1.17}$$

To ensure the condition (ii), ensuring that any task completing after $d-a$ completes after or at d is not sufficient. Indeed, for such a task j, reducing t'_j while satisfying the inequality (Q2') associated with $\{j\}$, *i.e.* $t'_j\geqslant p_j$, task j can become early if $p_j\leqslant a$. In this case, the induced penalty does not necessarily decrease. Figure 1.10 illustrates the extreme case of this phenomenon, that is when $a=d$, $E(\delta)=\emptyset$ (then (1.15) is satisfied) and tasks overlap each other to be on-time, while being in $T(\delta)$.

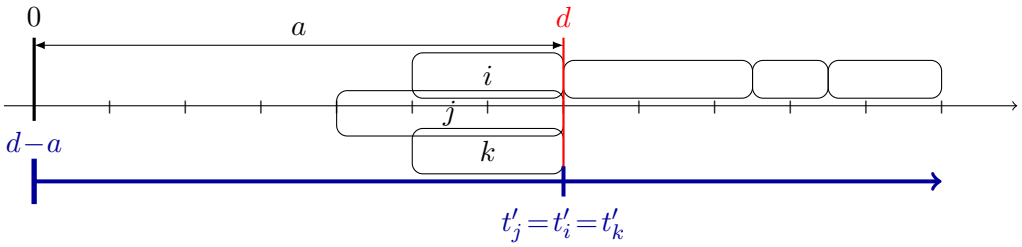


Figure 1.10: An unfeasible schedule when $a=d$

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, provided that it is not the first one. Therefore, in order to identify the first task j_0 completing after $d-a$, we introduce boolean variables $(\gamma_j)_{j\in J}$ along with the following inequalities

$$\sum_{j\in J}\gamma_j = 1 \tag{1.18}$$

$$\forall j\in J,\qquad \delta_j\leqslant 1-\gamma_j \tag{1.19}$$

$$\forall j\in J,\qquad t'_j\leqslant p_j+(1-\gamma_j)\,(p(J)-p_j) \tag{1.20}$$

More precisely, inequalities (1.18 – 1.19) ensure that γ designates one and only one task i_0 among those completing after $d-a$, *i.e.* among $T(\delta)$; while inequalities (1.20) ensure that i_0 is the first one, *i.e.* $i_0 = j_0$. Indeed, the latter ensure that $t'_{i_0} \leqslant p_{i_0}$, and since $t'_{i_0} \geqslant p_{i_0}$ by inequality (Q2') associated with the singleton $\{i_0\}$, we deduce that $t'_{i_0} = p_{i_0}$. Then, for each task j such that $\delta_j = 0$, the inequality (Q2') associated with a pair $\{i_0, j\}$ gives $\cancel{t'_{i_0} p_{i_0}} + t_j\, p_j \geqslant \cancel{p^2_{i_0}} + p_j^2 + p_{i_0}\, p_j$, which suffices to prove that task j completes after i_0, *i.e.* $t_j \geqslant p_{i_0} + p_j$. the following lemma sums up these results.

Lemma 1.15

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* (1.18)-(1.19) $\Leftrightarrow \exists i_0 \in T(\delta)$, $\gamma = \mathbb{I}_{i_0}$
(ii) ***If*** *(i) holds and* t', δ, X *satisfy* $(X.1)$-$(X.4)$, (1.20) *and (Q2′),*
then $t'_{i_0} = p_{i_0}$ *and* $\forall j \in T(\delta)$, $j \neq i_0 \Rightarrow t'_j \geqslant t'_{i_0} + p_j$.

Thanks to γ which identifies j_0, we can now ensure that j_0 does not complete before d, (*i.e.* $d - a + t'_{j_0} \geqslant d$, that is $a \leqslant t'_{j_0}$ or even $a \leqslant p_{j_0}$ since $p_{j_0} = t'_{j_0}$) using the following inequalities.

$$\forall j \in J, \qquad a \leqslant p_j + (1-\gamma_j)\, d \tag{1.21}$$

Yet again, these inequalities are valid since $d - a \geqslant 0$. Indeed, for a task $j \neq j_0$, $\gamma_j = 0$ and this inequality gives $a \leqslant p_j + d$.

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

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

$$\forall j \in J, \qquad b_j \geqslant 0 \tag{1.22}$$
$$\forall j \in J, \qquad b_j \leqslant a \tag{1.23}$$
$$\forall j \in J, \qquad b_j \leqslant \delta_j\, d \tag{1.24}$$
$$\forall j \in J, \qquad b_j \geqslant a - (1-\delta_j)\, d \tag{1.25}$$

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

Lemma 1.16

Let $(a, b, \delta) \in \mathbb{R} \times \mathbb{R}^J \times \{0,1\}^J$.
a, b *and* δ *satisfy inequalities* (1.22)-(1.25) $\Leftrightarrow b = a\,\delta$.

Then the total penalty of a schedule encoded by (e', t', a, b) is given by the linear function $h_{\alpha,\beta}$ defined as follows.

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

Note that, provided that the schedule satisfies the non-overlapping constraint, the function $h_{\alpha,\beta}$ provides the total penalty, even for a schedule admitting two different encodings, (*i.e.* for a d-schedule with at least one tardy task). To enunciate formally this result, let us introduce θ'_d (resp. $\widetilde{\theta'_d}$) the function which gives the first (resp. second) (e', t', a, b) encoding of a schedule from its completion time vector C. Functions θ'_d and $\widetilde{\theta'_d}$ are defined as follows.

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

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

Note that if C encodes a schedule with a straddling task, both encodings coincide, *i.e.* $\theta_d(C) = \widetilde{\theta'_d}(C)$, since $E(C) = \widetilde{E}(C)$ and $T(C) = \widetilde{T}(C)$. The following lemma gives the link between the objective function $h_{\alpha,\beta}$ and the previous objective function $f_{\alpha,\beta,d}$.

Lemma 1.17

*Let $C \in \mathbb{R}^J$. **If** C satisfies (1.1), **then** $h_{\alpha,\beta}\big(\theta'_d(C)\big) = h_{\alpha,\beta}\big(\widetilde{\theta'_d}(C)\big) = f_{\alpha,\beta,d}\big(\theta'_d(C)\big)$.*

- *Formulation F^4*

Let us first introduce the space of the variables introduced so far.

$$\mathbb{E} = \mathbb{R}^J \times \mathbb{R}^J \times [0,1]^J \times \mathbb{R}^{J^<} \times \mathbb{R} \times \mathbb{R}^J \times [0,1]^J$$

Now we can describe the polyhedron defined by the linear inequalities introduced so far.

$$\boldsymbol{P^4} = \left\{ (e', t', \delta, X, a, b, \gamma) \in \mathbb{E} \;\middle|\; \begin{array}{l} (1.5\text{'}-1.8\text{'}),\ (X.1-X.4),\ (1.15-1.17),\ (1.21-1.22), \\ (1.18-1.21),\ (\text{Q1'}) \text{ and } (\text{Q2'}) \text{ are satisfied} \end{array} \right\}.$$

Note that this polyhedron depends on d, in addition to p, but neither on α nor on β. Inequalities (Q1') and (Q2') require the same separation algorithm as for (Q1) and (Q2), which will be developed in Section 2.1.

We introduce the operator $\mathrm{int}_{\delta,\gamma}$, which only keeps points with integer δ and γ. of any set $V \subseteq \mathbb{E}$.

$$\mathrm{int}_{\boldsymbol{\delta,\gamma}}(\boldsymbol{V}) = \Big\{ (e', t', \delta, X, a, b, \gamma) \in V \;\Big|\; \delta \in \{0,1\}^J,\ \gamma \in \{0,1\}^J \Big\}$$

For the sequel, we also introduce $\pi'_{e,t,a,b}$ the projection of the whole variable space $\mathbb{E}$ to the (e, t') variable space as follows.

$$\boldsymbol{\pi'_{e,t,a,b}} = \begin{pmatrix} \mathbb{E} & \longrightarrow & \mathbb{R}^J \times \mathbb{R}^J \\ (e', t', \delta, X, a, b, \gamma) & \longmapsto & (e', t', a, b) \end{pmatrix}$$

Finally, our formulation for CDDP is the following.

$$F^4 : \min_{(e',t',\delta,X,a,b,\gamma) \in \mathrm{int}_{\delta,\gamma}\left(\mathrm{extr}\, P^4\right)} h_{\alpha,\beta}(e, t, a, b)$$

The following section shows the validity of F^4 following the same lines as Section 1.3.2 for F^3.

1.4.3 Validity of Formulation F^4

Let us start by summing up how we handle the non-overlapping[4] in Formulation F^4. Thanks to the natural variables e' and t', ensuring the non-overlapping constraint reduces to ensuring the non-negativity and non-overlapping constraints for two subsets of tasks. In contrast with Formulation F^3 where these two subsets are the early and the tardy tasks (*Cf.* Lemma 1.9), in Formulation F^4, the subsets to consider depend on the occurrence of a straddling or an on-time task, as detailed in the following lemma.

Lemma 1.18

Let $C \in \mathbb{R}^J$. Let $(e', t', a, b) \in \mathbb{R}^J \times \mathbb{R}^J \times \mathbb{R} \times \mathbb{R}^J$.

(i) ***If*** $(e', t', a, b) = \theta'_d(C)$, ***then*** *the following equivalence holds.*

$$\left.\begin{array}{r} C \text{ satisfies (1.1)} \\ \forall j \in J,\ C_j \leqslant d-a \text{ or } C_j \geqslant (d-a)+p_j \end{array}\right\} \Leftrightarrow \begin{cases} (e'+p)_{/E(C)} & \text{satisfies (1.0) and (1.1)} \\ t'_{/T(C)} & \text{satisfies (1.0) and (1.1)} \end{cases}$$

(ii) ***If*** $(e', t', a, b) = \widetilde{\theta'_d}(C)$, ***then*** *the following equivalence holds.*

$$\left.\begin{array}{r} C \text{ satisfies (1.1)} \\ \forall j \in J,\ C_j \leqslant d-a \text{ or } C_j \geqslant (d-a)+p_j \end{array}\right\} \Leftrightarrow \begin{cases} (e'+p)_{/\widetilde{E}(C)} & \text{satisfies (1.0) and (1.1)} \\ t'_{/\widetilde{T}(C)} & \text{satisfies (1.0) and (1.1)} \end{cases}$$

The following theorem establishes that a feasible schedule, under some assumptions, is encoded by an integer point of P^4. In particular a d-or-left-block is encoded by an integer point of P^4.

Theorem 1.19

Let $C \in \mathbb{R}^J$ satisfying (1.0) *and* (1.1).

(i) ***If*** *there exists $j_s \in J$ such that $C_{j_s} - p_{j_s} < d < C_{j_s}$,*
and if *$\forall j \in J,\ d - p(J) \leqslant C_j - p_j$ and $C_j \leqslant C_{j_s} - p_{j_s} + p(J)$,*
then *there exists $(e', t', \delta, X, a, b, \gamma) \in \mathrm{int}_{\delta,\gamma}(P^4)$ such that $\theta'_d(C) = (e', t', a, b)$.*

(ii) ***If*** *there exists $j_t \in J$ such that $C_{j_t} = d$,*
and if *$\forall j \in J,\ d - p(J) \leqslant C_j - p_j$ and $C_j \leqslant C_{j_t} - p_{j_t} + p(J)$,*
then *there exists $(e', t', \delta, X, a, b, \gamma) \in \mathrm{int}_{\delta,\gamma}(P^4)$ such that $\widetilde{\theta'_d}(C) = (e', t', a, b)$.*

Proof: Let us start by proving *(i)*.

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

We will prove that $Y \in \mathrm{int}_{\delta,\gamma}(P^4)$. Since $\delta \in \{0,1\}^J$ and $\gamma \in \{0,1\}^J$ by construction, we have to show that Y satisfies all the inequalities defining P^4, *i.e.* $Y \in P^4$.

Note that the definition of δ ensures that $E(\delta) = \mathsf{E}(C)$ and $T(\delta) = \mathsf{T}(C)$, which allows the notation E and T for the sake of brevity. By Lemma 0.5*(i)*, the definition of X ensures that inequalities $(X.1 - X.4)$ are satisfied. By Lemma 1.15*(i)*, the definition of γ ensures that inequalities (1.18 – 1.19) are satisfied, since $j_s \in T$. By Lemma 1.16, inequalities (1.22 – 1.25) are satisfied, since $b = a\,\mathbb{I}_{\mathsf{E}(C)} = a\,\delta$, by definition of θ'_d.

Since C encodes a feasible schedule, the straddling task j_s satisfies $C_{j_s} - p_{j_s} = \min_{j \in \mathsf{T}(C)} (C_j - p_j)$.
Then $a = d - (C_{j_s} - p_{j_s})$, by definition of θ'_d.

We also have $0 \leqslant C_{j_s} - p_{j_s} < d$, then $0 < a \leqslant d$. Thus inequality (1.17) is satisfied, and for any task $j \neq j_s$, $\gamma_j = 0$ then $a \leqslant d + p_j = (1 - \gamma_j)\, d + p_j$. 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 (1.16) holds, since $E = E(\delta)$. Moreover, task j_s

[4]This paragraph, along with Lemma 1.18, is not placed before, since the complete encoding and the functions θ'_d and $\widetilde{\theta'_d}$ must be already introduced for a correct enunciation.

completes after d, *i.e.* $C_{j_s} > d$, thus we get $a = p_{j_s} + (d - C_{j_s}) < p_{j_s} = p_{j_s} + (1-\gamma_{j_s})\, d$, since $\gamma_{j_s} = 1$. We deduce that inequalities (1.21) are satisfied.

Inequalities (1.5') and (1.7'), are satisfied by construction of e' and t' (*Cf.* definition of θ'_d).

Let us prove that Y satisfies inequalities (1.5' – 1.8'), (1.15), and (1.20) by considering separately the inequalities indexed by $j \in E$ of those indexed by $j \in T$. We will say "the inequality $(\bullet)$" meaning implicitely "the inequality $(\bullet)$ indexed by j".

$\rightarrow$ For a task j in E, $C_j \leqslant C_{j_s} - p_{j_s} = d - a$ since j and j_s do not overlap, then $e'_j = d - a - C_j$ and $t'_j = 0$. Inequality (1.8') is thus satisfied, along with (1.20) since $p_j + (1-\gamma_j)\,(p(J) - p_j) = p(J) \geqslant 0 = t'_j$.
By assumption $C_j \geqslant d - p(J) + p_j$, thus $e'_j \leqslant d - a - (d - p(J) + p_j) \leqslant p(J) - p_j$, and inequality (1.6') is also satisfied. Moreover, $d - e'_j - p_j\delta_j = a + C_j - p_j$, and by non-negativity constraint $C_j - p_j \geqslant 0$, thus $d - e'_j - p_j\delta_j \geqslant a$ and inequality (1.15) is satisfied .

$\rightarrow$ 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 inequality (1.6') is thus satisfied. Moreover, $d - e'_j - p_j\delta_j = d \geqslant a$, then inequality (1.15) is satisfied for j. By assumption $C_j \leqslant (C_{j_s} - p_{j_s}) + p(J) = (d - a) + p(J)$, thus $t'_j = C_j - (d - a) \leqslant p(J)$. We deduce that the inequality (1.8') is also satisfied, along inequality (1.20) if $j \neq j_s$, since $p_j + (1-\gamma_j)\,(p(J) - p_j) = p(J)$ in this case. If $j = j_s$, inequality (1.20) is also satisfied, since $pj_s + (1-\gamma_{j_s})\,(p(J) - p_{j_s}) = p_{j_s} = C_{j_s} - (d - a) = t'_{j_s}$.

Since C encodes a feasible schedule, C satisfies (1.1). Moreover, by definition of θ'_d, $d - a$ is the starting tie of a task (namely j_0). We deduce that no task starts before $d - a$ and completes after $d - a$, *i.e.* $\forall j \in J,\ C_j \leqslant d - a$ or $C_j \geqslant (d - a) + p_j$. Then Lemma 1.18*(i)* ensures that $(e' + p)_{/E}$, along with $t'_{/T}$, satisfies (1.0) and (1.1). Applying Property 1.1 to these two vectors, we deduce that they satisfy (Q0), and using Lemma 1.10, that e', δ, X satisfy (Q1') and t', δ, X satisfy (Q2').

Thus, Y belongs to $\mathrm{int}_{\delta,\gamma}(P^4)$ and *(i)* is proved

Rewriting the proof by replacing θ'_d by $\widetilde{\theta'_d}$, $\mathsf{E}(C)$ by $\widetilde{\mathsf{E}}(C)$, $\mathsf{T}(C)$ by $\widetilde{\mathsf{T}}(C)$, 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 for inequality (1.21) indexed by j_t: in this case $C_{j_t} = d$, and $a = d + (p_{j_t} - C_{j_t}) = p_{j_t} = p_{j_t} + (1-\gamma_{j_t})\, d$. □

The following theorem establishes that an optimal solution of formulation F^4 encodes a solution for CDDP, and even a dominant solution since it encodes a d-or-left-block.

Theorem 1.20

Let assume *that* $\alpha \in \mathbb{R}_+^{*\,J}$. *Let* $Y^* = (e', t', \delta, X, a, b, \gamma) \in int_{\delta,\gamma}(P^4)$.
If $Y^* \in \mathrm{extr}(P^4)$ ***and*** (e', t', a, b) *minimizes* $h_{\alpha,\beta}$ *on* $\pi'_{e,t,a,b}\,(int_{\delta,\gamma}(P^4))$,
then Y^* *encodes a d-or-left-block, by* θ'_d *or* $\widetilde{\theta'_d}$.

Proof: Let us set, for any task j in J, $C^*_j = (d - a) - e'_j + t'_j$. The first step of the proof is to show that C^* gives the completion times of the schedule encoded by Y^* using θ'_d or $\widetilde{\theta'_d}$.

– *Proving that* $(e', t', a, b) = \theta'_d(C^*)$ *or* $(e', t', a, b) = \widetilde{\theta'_d}(C^*)$

First we derive from inequalities (1.5' – 1.8') 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 Y^* satisfies $(X.1 - X.4)$, (1.18 – 1.20) and (Q2'), Lemma 1.15 ensures that there exists $j_0 \in T(\delta)$ such that $\gamma = \mathbb{I}_{\{j_0\}}$, $t'_{j_0} = p_{j_0}$, and $\forall j \in T(\delta),\ j \neq j_0,\ t'_j \geqslant t'_{j_0} + p_j$. Since j_0 is in $T(\delta)$, $e'_{j_0} = 0$, hence $C^*_{j_0} - p_{j_0} = d - a$. Then for any other task j in $T(\delta)$, we have

$$C^*_j - p_j = (C^*_{j_0} - p_{j_0}) + t'j - p_j \geqslant (C^*_{j_0} - p_{j_0}) + t'_{j_0} = C^*_{j_0} > C^*_{j_0} - p_{j_0}.$$

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

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 (1.21) gives $a \leqslant p_{j_0}$, thus $C^*_j > d$. We deduce that $T(\delta) \backslash \{j_0\} \subseteq \mathsf{T}(C^*)$. Conversely, for a task j in $\mathsf{T}(C^*)$, $C^*_j > d$,

which is equivalent to $t'_j - e'_j > a$. Since $a \geqslant 0$ by inequality (1.17), $t'_j > e'_j$, which would be impossible if j was in $E(\delta)$, according to inequalities (1.5') and (1.8'). We deduce that $\mathsf{T}(C^*) \subseteq T(\delta)$. Two cases have to be considered.

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

For the remainder of the proof, we assume that we are in the first case. Then E (resp. T) will denote $E(\delta) = \mathsf{E}(C^*)$ (resp. $T(\delta) = \mathsf{T}(C^*)$), and we will use the encoding θ'_d. To handle the second case, it suffices to replace $\mathsf{E}(C^*)$ by $\widetilde{\mathsf{E}}(C^*)$, $\mathsf{T}(C^*)$ by $\widetilde{\mathsf{T}}(C^*)$, and θ'_d by $\widetilde{\theta'_d}$ to apply Lemma 1.18*(ii)* instead of Lemma 1.18*(i)*, and, in the third step, to use that j_0 is the on-time task.

We can rewrite δ as $\mathbb{I}_{\mathsf{E}(C^*)}$, and thus b as $a\,\mathbb{I}_{\mathsf{E}(C^*)}$, since $b = a\,\delta$ by inequalities (1.22–1.25) and Lemma 1.16. Using inequalities (1.5'-1.8'), 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 $(e', t', a, b) = \theta'_d(C^*)$, that is that C^* and (e', t', a, b) encode the same schedule, which will be denoted by $\mathcal{S}^*$.

– *Proving that $\mathcal{S}^*$ is feasible*

We have to prove that C^* satisfies (1.0) and 1.1).
For a task j in E, inequality (1.15) ensures that $p_j \leqslant d-a-e'_j = C^*_j$. For a task j in T, inequality (1.15) ensures that $a \leqslant d$, then $C^*_j = d-a+t'_j \geqslant t'_j$. Moreover, from inequality (Q2') associated with $\{j\}$, we have $t'_j \geqslant p_j$ thus C^* satisfies (1.0).

To show that C^* satisfies (1.1) using Lemma 1.18*(i)*, it remains to show that vectors $(e'+p)_{/E}$ and $t'_{/T}$ satisfy (1.0) and (1.1). Since inequalities (Q1') and (Q2') are satisfied, we know from Lemma 1.10 that $(e'+p)_{/E}$ and $t'_{/T}$ satisfy inequalities (Q0). First, these inequalities for the singletons ensure that both vectors satisfy (1.0). Second, inequalities (Q0) allow us to show that both vectors satisfy (1.1) as follows

Let us assume that $(e'+p)_{/E}$ does not satisfy (1.1). 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.

- → If $e'_j+p_j \geqslant p(J)$, then $e'_j+p_j \geqslant p(E)$. Applying Lemma 1.5 to $(e'+p)_{/E}$, we can construct a vector $e'^- \in \mathbb{R}^J$ which is component wise smaller than e' and such that $(e'^-+p)_{/E}$ satisfies (Q0). One an check that $Y^- = (e'^-, t', \delta, X, a, b, \gamma)$ is in $\mathrm{int}_{\delta,\gamma}(P^4)$ and $h_{\alpha,\beta}(e'^-, t', a, b) < h_{\alpha,\beta}(e', t', a, b)$ since $\alpha \in {\mathbb{R}^*_+}^J$. A contradiction since (e', t', a, b) minimizes $h_{\alpha,\beta}$ on $\pi'_{e,t,a,b}\left(\mathrm{int}_{\delta,\gamma}(P^4)\right)$.
- → If $e'_j+p_j = d-a$, we can derive the same contradiction since $d-a \geqslant p(E)$ from inequality (1.16).
- → If $e'_j+p_j < p(J)$ and $e'_j+p_j < d-a$, according to Remark 1.4, we can force the parameter ε to be smaller than $\min(p(J)-(e'_j+p_j), d-a-(e'_j+p_j)) > 0$ when applying Lemma 1.3 to $(e'+p)_{/E}$. This way, we obtain two vectors $e'^{+-} \in \mathbb{R}^J$ and $e'^{-+} \in \mathbb{R}^J$ such that $(e'^{+-}+p)_{/E}$ and $(e'^{-+}+p)_{/E}$ both satisfy (Q0), an such that $Y^{+-} = (e'^{+-}, t', \delta, X, a, b, \gamma)$ and $Y^{-+} = (e'^{-+}, t', \delta, X, a, b, \gamma)$ are in $\mathrm{int}_{\delta,\gamma}(P^4)$. Yet Y^* is the middle point of the segment $[Y^{+-}, Y^{-+}]$. this contradicts the extremality of Y^*.

Similarly, let us assume that $t'_{/T}$ does not satisfy (1.1). Then there exist two tasks i and j in T such that $t'_i \leqslant t'_j < t'_i+p_j$. By Lemma 1.15, $\forall k \in T(\delta)$, $k \neq j_0 \Rightarrow t'_k \geqslant t'_{j_0}+p_k$, we deduce that $i \neq j_0$. Then for tasks i and j, inequalities (1.8') and (1.20) are equivalent, and t'_i and t'_j are only bounded from above by $p(J)$. Then two cases have to be considered.

- → If $t'_j \geqslant p(J)$, then $t'_j \geqslant p(T)$. Applying Lemma 1.5 to $(e'+p)_{/E}$ we can derive a contradiction to the minimality of (e', t', a, b) on $\pi'_{e,t,a,b}\left(\mathrm{int}_{\delta,\gamma}(P^4)\right)$.
- → If $t'_j < p(J)$, applying Lemma 1.3 we can derive a contradiction to the extremality of Y^*.

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

– *Proving that $\mathcal{S}^*$ is a d-or-left-block.*

Let us denote by $\mathscr{S}$ the set of blocks satisfying at least one condition among: - the starting time is 0,
- there is an on-time task,
- $\alpha(E)=\beta(T)$.

From the proof of Lemma 0.2, $\mathscr{S}$ is a strictly dominant. Therefore, we can show that $\mathcal{S}^*$ necessary belongs to $\mathscr{S}$ as we proved that $\mathcal{S}^*$ necessary was a d-block in Theorem 1.13's proof. Indeed, if $\mathcal{S}^*$ does not belongs $\mathscr{S}$, there exists $\widehat{\mathcal{S}}$ having a lower penalty, an according to Theorem 1.19, this schedule corresponds to a point in $\text{int}_{\delta,\gamma}(P^4)$ which contradicts the minimality of (e', t', a, b).

Conversely, the set of d-or-left-blocks is not a strictly dominant set. To show that $\mathcal{S}^*$ is a d-or-left-block, we then use the extremality of Y^* as follows. Let us assume that $\mathcal{S}^*$ is not a d-or-left-block. We necessarily have $\alpha(E)=\beta(T)$ since $\mathcal{S}^*\in\mathscr{S}$. That implies that $E\neq\emptyset$ since $\beta\neq 0$ (resp. $T\neq\emptyset$ since $\alpha\neq 0$). Then in $\mathcal{S}^*$, there is a straddling task (necessarily it is j_0) and a tasks scheduled right before it. The first completes at time $d-a+p_{j_0}$ while the second completes at time $d-a$, then we have $p_{j_0}-a>0$ and $a>0$. Moreover, denoting by s the starting time of $\mathcal{S}^*$, we have $s>0$. Then $\varepsilon=\frac{1}{2}\min(p_{j_0}-a,\ a,\ s)$ is positive. Setting $a^-=a-\varepsilon$ and $Y^-=(e', t', \delta, X, a^-, a^-\delta, \gamma)$ (resp. $a^+=a+\varepsilon$ and $Y^+=(e', t', \delta, X, a^+, a^+\delta, \gamma)$), Y^- (resp. Y^+) encodes using θ'_d the schedule obtained by shifting backward (resp. forward) the whole block by ε time unit. By definition of ε, Y^- (resp. Y^+) still satisfies inequalities (1.15), (1.21), (1.16) and (1.17), thus $Y^-\in P^4$ (resp. $Y^+\in P^4$). Since Y^* is the middle of $[Y^-, Y^+]$, that contradicts the extremality of Y^*.

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

• *Dealing with zero unit earliness penalties*

If some tasks have a zero unit earliness penalty, F^4 provides a vector $Y^*=(e', t', \delta, X, a, b, \gamma)$ 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 Y^* does not ensure that these tasks are well spread out (in this context Lemma 1.5 cannot be applied). However, the minimality of Y^* 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 (1.16), 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 $h_{\alpha,\beta}(Y^*)$, regardless of their order. We deduce that $\mathcal{S}$ is an optimal schedule.

The following theorem establishes that CDDP reduces to solving formulation F^4. We omit the proof since it follows the same lines as the one of Theorem 1.14.

Theorem 1.21

Any optimal d-or-left-block, is encoded by a vector minimizing $h_{\alpha,\beta}$ on $\text{int}_{\delta,\gamma}\big(\text{extr}\, P^4\big)$.
Conversely, any vector minimizing $h_{\alpha,\beta}$ on $\text{int}_{\delta,\gamma}\big(\text{extr}\, P^4\big)$, encodes an optimal d-or-left-block.

Formulations F^3 and F^4 are built and then proved to be valid using the same ideas. We recall below these common ideas as it can be seen as a guideline to formulate a problem using natural variables and non-overlapping inequalities. Of course, this method is not applicable to any scheduling problem, but the following section gives insights of such possible extensions.

1.4.4 A guideline to provide a formulation using natural variables

1. Establish the **dominance properties** standing for the considered problem. Use them to consider only a subset of schedules as solution set.

2. For such a schedule, choose how it can be cut in order to **decompose** the global non-overlapping constraint into independent local non-overlapping constraints. The time axis is then decomposed into several half-axis. Each of them is characterized by a **reference point**, a direction (left or right, *i.e.* before or after), and a machine (if several). Note that the total penalty has to decrease (or at least not raise) when a task gets closer to its reference point.

3. If the reference point depends on the schedule (let us talk about **floating reference point** in this case), introduce variables along with inequalities to manage this floating reference point.

4. Introduce boolean variables to encode the task **partition** induced by the decomposition.

5. Introduce for each part of the partition a family of **natural variables** measuring the distance to the reference point.

6. Add inequalities to ensure that the natural variables are **consistent** with the partition variables.

7. Add **non-overlapping inequalities** for each family of natural variables, that is for each part.

8. If needed, add **linearization** variables, along with the needed linearization inequalities, so as to obtain a linear objective function.

$\Longrightarrow$ The problem is then to minimize a linear function f over the integer extreme points of a polyhedron.

9. Check that each dominant schedule is encoded by an integer point of the polyhedron.

10. Check that each integer extreme point of the polyhedron encodes a feasible schedule and even a dominant schedule.

11. Show that each optimal schedule is encoded by an integer extreme point of the polyhedron minimizing f, and conversely that such a point encodes an optimal schedule.

1.5 Using natural variables and non-overlapping inequalities for related problems

In this section, we consider just-in-time scheduling problems similar to common due date problems, where a formulation using natural variables and non-overlapping inequalities could be provided using the ideas presented in the previous section. We give neither complete formulation nor validity proof, but only give some hints.

1.5.1 Slight modifications of the penalty function

Let us consider again the situation of the joiner who wants to send its daily production at 4:00 pm (*Cf.* Page 12), and imagine other costs for the extra-deliveries, that is the delivery of furniture completing after 4:00 pm.

• *Extra-delivery cost with a setup cost*

Let us assume that an extra-delivery does not only induce a cost proportional to the tardiness but also a setup cost σ_j which depends on the piece of the furniture j which has to be delivered. In this case, the total penalty is the following.

$$\sum_{j \in E} \alpha_j \, E_j + \sum_{j \in T} (\beta_j \, T_j + \sigma_j)$$

Following the same line as for UCDDP (resp. CDDP), one can show that d-blocks (resp. d-or-left-blocks) are dominant for this problem if the due date is unrestrictive (resp. even if it is not the case). Moreover, using the already introduced variables, the objective function can be written as follows.

$$\sum_{j \in J} \alpha_j \, e_j + \beta_j \, t_j + \sigma_j \, (1 - \delta_j)$$

This is a linear function, which is in addition a non-increasing function of each variable e_j or t_j. That suffices to use Formulation F^3 (resp. F^4) for this problem, changing only the objective function.

• *Extra-delivery cost reduced to a setup cost*

Let us assume that the cost of an extra-delivery is only a setup cost, that is a cost that does not depend on the completion time. In this case, one can show that d-blocks are dominant, even if the due date is unrestrictive. Indeed, if a straddling task j_s occurs in a block, right-shifting the whole block so as to start j_s at time d does not increase the total penalty since it does not increase the earliness penalty, there is no tardiness penalty, and the penalty due to the tardy task does not change. Therefore, no matter the value of the due date, Formulation F^3 can be adapted for this problem using the following objective function.

$$\sum_{j \in J} \alpha_j \, e_j + \sigma_j \, (1 - \delta_j)$$

Variables $(t_j)_{j \in J}$, along with non-overlapping inequalities (Q2) can be removed, since tardy tasks can be arbitrarily scheduled after d with no impact on the total penalty.

• *Extra-delivery by batch*

Let us now assume that the joiner chooses to deliver all the furniture completed after 4:00 pm at one blow, instead of delivering each piece of furniture by bike. The joiner has then to wait that the last piece of furniture completes. One can consider that the cost of this single extra batch delivery is proportional to the tardiness of the last task. The total penalty function can then be written as follows.

$$\sum_{j\in J}(\alpha_j\, E_j) + \lambda \left[\max_{j\in J} C_j - d\right]^+$$

Assuming that the due date is unrestrictive, one can show that the d-blocks are dominant. Therefore, the positive part on the last term can be removed, since necessarily $\max_{j\in J} C_j \geqslant d$. In order to obtain a linear objective function, we have to introduce a single continuous variable t_{max} to represent $\max_{j\in J} t_j = \max_{j\in J} C_j - d$, along with the following single constraint.

$$t_{max} \geqslant \sum_{j\in J} p_j\,(1-\delta_j)$$

Hence, the objective function is the the following linear function.

$$\sum_{j\in J}(\alpha_j\, e_j) + \lambda\, t_{max}$$

As previously, variables $(t_j)_{j\in J}$, along with non-overlapping inequalities (Q2) can be removed, since tardy tasks can be arbitrarily sequence after d, as long as they form a block starting at d.

We proposed here only modification on the penalty due to tardy tasks, however, we could similarly manage other penalties for the early tasks, provided they could be linearly expressed. Indeed, variables $(e_j)_{j\in J}$ and $(t_j)_{j\in J}$ are almost independent once variables δ_j have fix the early-tardy partition.

1.5.2 From one machine to multi-machine

• *What can model a multi-machine setting*

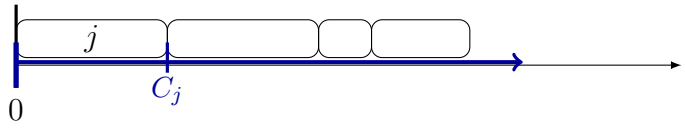
Let us now assume that the joinery is divided into multiple workshops. We then have to decide in which workshop each piece of furniture will be produced. If the different workshops are not located in the same place, the delivery can start at different time in each one: for example at 4:00 pm in the one while at 5:00 pm in the other which is located closer to the customers. Moreover, the congestion in one workshop is independent to the other. Therefore, multiple workshops are modeled by multiple machines, denoted by $[1..m]$, and having their own due date denoted by $(d^k)_{k\in[1..m]}$. One can even consider unit earliness and tardiness penalties depending on the machine. For example in a more spacious workshop, the unit earliness penalty can be lower.

Let us assume that all the due dates are unrestrictive. In this multi-machine setting, one can show a dominance property equivalent to the V-shaped d-block dominance property for UCDDP. Indeed, the schedules such that, for each machine $k \in [1..m]$, the tasks scheduled on k form a V-shaped d^k-block, are dominant. Therefore, Formulation F^3 can be adapted to this problem.

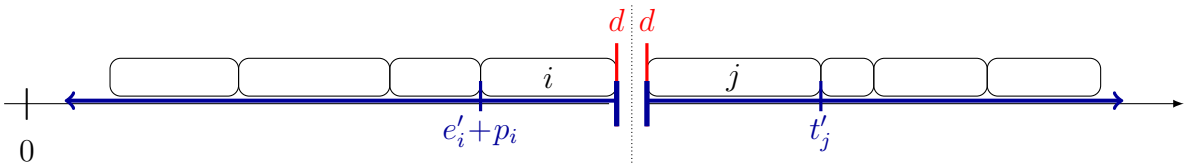
• *Correspondence between natural variables families and half-axes*

Each natural variable family, along with the associated non-overlapping inequalities, can be seen as a way to manage tasks placed on an half-axis, whose origin is attractive for tasks, *i.e.* the closer tasks are placed to the origin, the lower is the penalty they induced.

As represented below, variables $(C_j)_{j\in J}$, along with Queyranne's non-overlapping inequalities (Q0), correspond to the half-axis $[0, +\infty[$.

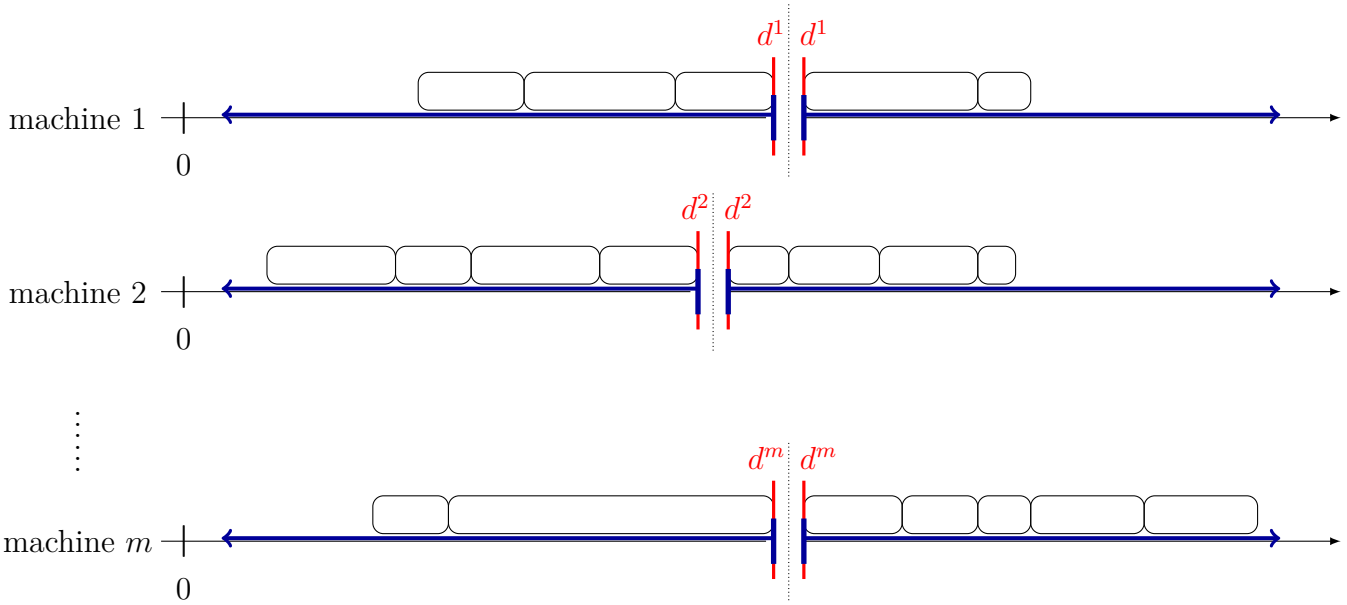


As represented below, variables $(e_i)_{i\in E(\delta)}$, or more precisely $(e_i+p_i)_{i\in E(\delta)}$ (resp. $(t_j)_{j\in T(\delta)}$), along with non-overlapping inequalities (Q1) (resp. (Q2)), correspond to the half-axis $]-\infty, d[$ (resp. $[d, +\infty[$).



Note that the passage from one axis to two axes require the introduction of a binary variable for each task, indicating to which axis the task belongs. That allow to write non-overlapping inequalities taking into account only the appropriate tasks for each axis, *i.e.* only those belonging to this axis. It is exactly what allows δ_j on page 60. To be exact, an additional variable, along with four inequalities, must be introduced for each pair of tasks, in order to produce *linear* non-overlapping inequalities. It is exactly the role of X variables on page 60.

As represented below, with m machines, $2m$ half-axes have to be considered: two for each machine, one for the early tasks, the other for the tardy ones.



Hence, we have to introduce $2\,m-1$ binary variables for each task, $2\,m$ variables along with four inequalities for each pair of tasks. We also introduce $2\,m$ families of natural variables indexed by J, *i.e.* one continuous variable for each task and each half-axis, along with the corresponding non-overlapping inequalities.

The resulting formulation, which is similar to F^3, was the subject of the internship of Alexandre Heintzmann, advised by Pierre Fouillhoux and myself during summer 2019. [référence à venir]

1.5.3 From a common due date to a common due window

Let us forget the delivery problem and focus on the working day organization according to the daylight. The joiner prefers to work in daylight rather than artificial light, especially for finishing touches. If the time window $[d^{\sqsubset}, d^{\sqsupset}]$ represents the period of the day where there is enough sunlight, finishing touches for tasks completing in this time windows are not painful. Therefore, no penalty will occur for such tasks in the model. Conversely, finishing a furniture before the sunrise (resp. after the sunset) is painful. More precisely, the more it is early in the morning (resp. late at night) the more painful it is. Therefore, each task j completing before $d^{\sqsubset}$ (resp. after $d^{\sqsupset}$) will incur a penalty proportional to $d^{\sqsubset}-C_j$ (resp. $C_j-d^{\sqsupset}$). A priori, one can think to symmetrical earliness and tardiness unit penalties, *i.e.* $\forall j \in J,\ \alpha_j = \beta_j$, to reflects this situation, but taking into account the eye fatigue at the end of the day, one can also have $\forall j \in J,\ \alpha_j \leqslant \beta_j$. Finally, the objective function is the following.

$$\sum_{j \in J} \alpha_j \, [d^{\sqsubset} - C_j]^+ + \beta_j \, [C_j - d^{\sqsupset}]^+$$

Figure 1.11 represents the penalty incurred by a task j in function of its completion time C_j.

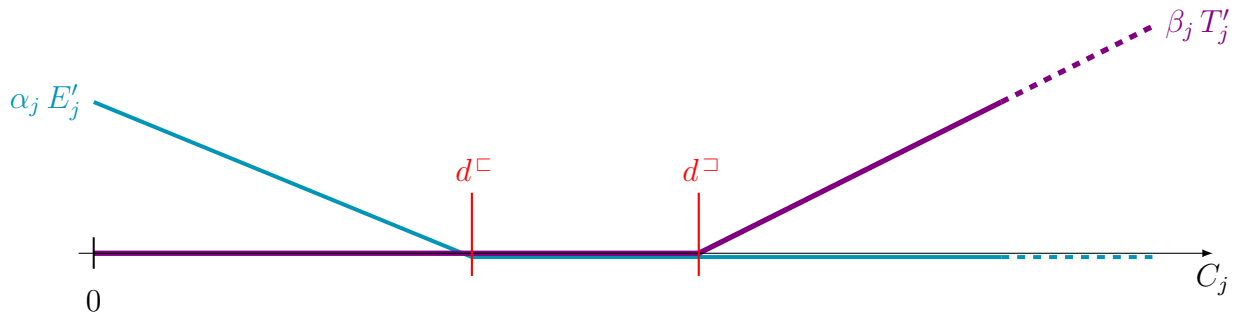


Figure 1.11: Earliness and tardiness penalties of j in function of C_j for a due window $[d^{\sqsubset}, d^{\sqsupset}]$

This problem is known in the literature as the common due window scheduling problem. Let us denote it by **CDWP** . Kramer and Lee [29] study CDWP in the case of task independent earliness and tardiness penalties, *i.e.* $\forall j \in J, \alpha_j = \alpha_0$ and $\beta_j = \beta_0$. They propose a dynamic programming algorithm in $\mathcal{O}(n\, d^{\sqsupset})$ that exactly solves CDWP. We consider a larger class of instances, since we allow task-dependent unit earliness and tardiness penalties. Therefore, the dominance properties announced in the following are less strong than the ones provided in [29].

The dominance properties for UCDDP and CDDP can be extended to this problem as proposed in Lemma 1.22, where definitions have been adapted as follows. For a given instance, a schedule is a **$d^{\sqsubset}$-or-$d^{\sqsupset}$-block** if it is a block and there exists a task completing at time $d^{\sqsubset}$ or a task completing at time d^2. A schedule is a **$d^{\sqsubset}$-or-$d^{\sqsupset}$-or-left-block** if it is a $d^{\sqsubset}$-or-$d^{\sqsupset}$-block or a left-block. A schedule is said **$\vee$-shaped** if the tasks completing before $d^{\sqsubset}$ are scheduled by non-decreasing α-ratio and the tasks *starting* after $d^{\sqsupset}$ are scheduled by non-increasing β-ratio. Moreover, for a given schedule, a task is said **early** (resp. **tardy**) if it completes before $d^{\sqsubset}$ (resp. after $d^{\sqsupset}$). Note that there can be some tasks which are neither early nor tardy.

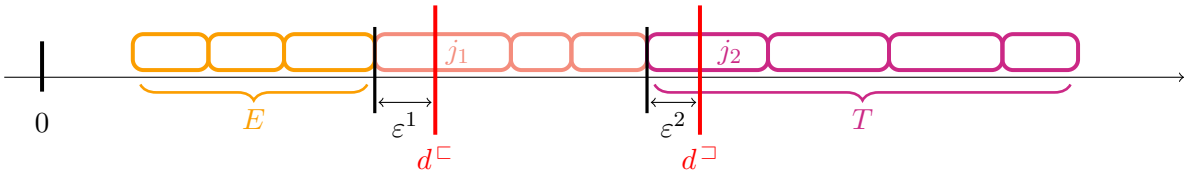
Lemma 1.22

Let $p \in \mathbb{R}_+^J$ and $(\alpha, \beta) \in \mathbb{R}_+^J \times \mathbb{R}_+^J$. Let $(d^{\sqsubset}, d^{\sqsupset}) \in {\mathbb{R}_+}^J \times {\mathbb{R}_+}^J$.
The set of blocks is strictly dominant for CDWP$(p, \alpha, \beta, d^{\sqsubset}, d^{\sqsupset})$.
The set of $d^{\sqsubset}$-or-$d^{\sqsupset}$-or-left-blocks is dominant for CDWP$(p, \alpha, \beta, d^{\sqsubset}, d^{\sqsupset})$.
Moreover, if $d^{\sqsubset} \geqslant p(J)$ the set of $d^{\sqsubset}$-or-$d^{\sqsupset}$-blocks is dominant for CDWP$(p, \alpha, \beta, d^{\sqsubset}, d^{\sqsupset})$
The set of $\vee$-shaped schedules is strictly dominant.

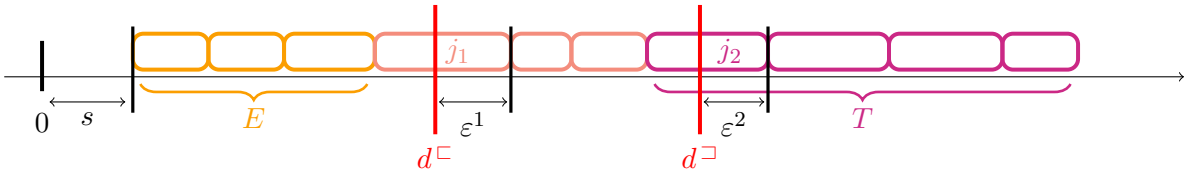
Proof: These properties can be shown using shifting and exchange arguments, as done for common due date problems in Section 0.2. However, we develop the proof of the $d^{\sqsubset}$-or-$d^{\sqsupset}$-or-left-block dominance property in order to convince the reader that we cannot ensure that an optimal schedule can be found among the $d^{\sqsubset}$-blocks (resp. $d^{\sqsupset}$-blocks), even if $d^{\sqsubset} \geqslant p(J)$. In other words, one cannot choose $d^{\sqsubset}$ or $d^{\sqsupset}$ as preferred reference point for dominant schedules.

Thanks to the block dominance properties, there exists an optimal schedule $\mathcal{S}^*$, encoded by the completion times $(C_j)_{j\in J}$, which is a block. Let us assume that $\mathcal{S}^*$ is not $d^{\sqsubset}$-or-$d^{\sqsupset}$-or-left-block. As done to prove the d-block dominance property on page 16, one can prove that $\mathcal{S}^*$ can neither completes before $d^{\sqsubset}$ nor starts after $d^{\sqsupset}$, by optimality. There exist necessarily a task $j_1 \in J$ such that $C_{j_1} - p_{j_1} < d^{\sqsubset} < C_{j_1}$, and a task $j_2 \in J$ such that $C_{j_2} - p_{j_2} < d^{\sqsubset} < C_{j_2}$. Possibly, $j_1 = j_2$.

→ If $\alpha(E) > \beta(T)$, let us set $\varepsilon^1 = d^{\sqsubset} - (C_{j_1} - p_{j_1})$, $\varepsilon^2 = d^{\sqsupset} - (C_{j_2} - p_{j_2})$, and $\varepsilon = \min(\varepsilon^1, \varepsilon^2)$. By definition of j_1 and j_2, $\varepsilon^1 > 0$ and $\varepsilon^2 > 0$, then $\varepsilon > 0$. By right-shifting the whole block by ε time units, the total penalty reduces by $\varepsilon(\alpha(E) - b(T)) > 0$. A contradiction since $\mathcal{S}$ is optimal.



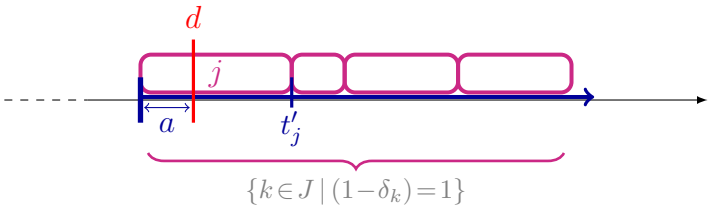
→ If $\alpha(E) < \beta(T)$ let us set $\varepsilon^1 = C_{j_1} - d^{\sqsubset}$, $\varepsilon^2 = C_{j_2} - d^{\sqsupset}$, s the starting time of $\mathcal{S}$, and $\varepsilon = \min(\varepsilon^1, \varepsilon^2, s)$. Since $\mathcal{S}$ is not a left-block, $s > 0$, and by definition of j_1 and j_2, $\varepsilon^1 > 0$ and $\varepsilon^2 > 0$, then $\varepsilon > 0$. By left-shifting the whole block by ε time units, the total penalty reduces by $\varepsilon\,(\beta(T) - \alpha(E)) > 0$. A contradiction since $\mathcal{S}$ is optimal.



→ If $\alpha(E) = \beta(T)$, there is an infinite number of optimal schedules which are not $d^{\sqsubset}$-or-$d^{\sqsupset}$-blocks: all those obtained by right-shifting the whole block by $\varepsilon < \min\,(d - (C_{j_1} - p_{j_1}),\ d^{\sqsupset} - (C_{j_2} - p_{j_2}))$ and all those obtained by left-shifting the whole block by $\varepsilon < \min(s,\ C_{j_1} - d^{\sqsubset},\ C_{j_2} - d^{\sqsupset})$. Indeed, such shifting operation induce no variation of the total penalty since $\alpha(E) = \beta(T)$. However, there also exists an optimal $d^{\sqsubset}$-or-$d^{\sqsupset}$-block since there is at least the one obtained by a $\min(s,\ C_{j_1} - d^{\sqsubset},\ C_{j_2} - d^{\sqsupset})$ left-shifting.

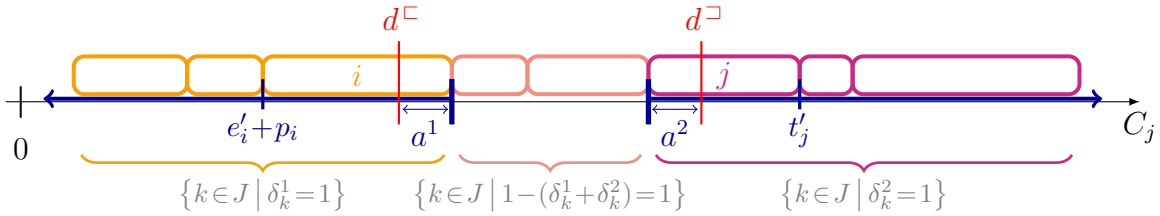
Finally, an optimal schedule can always be found among the $d^{\sqsubset}$-or-$d^{\sqsupset}$-or-left-blocks.
Moreover, if $d^{\sqsubset} \geqslant p(J)$, the previous min is not reached by s, and we can transform any optimal block into a $d^{\sqsubset}$-or-$d^{\sqsupset}$-block. □

In spite of the $d^{\sqsubset}$-or-$d^{\sqsupset}$-block dominance property, we cannot encode only schedules where tardy tasks starts at time $d^{\sqsupset}$, since there can be a $d^{\sqsupset}$-straddling task in all the optimal schedule. We are in the same situation as for CDDP with a restrictive due date, where the encoding had to manage schedules where the first tardy task starts before d. As represented below, to address this issue we proposed to introduce a new variable a so as to $d-a$ represents the starting time of the first tardy task.



Roughly speaking, the variable a allowed to make floating the origin of the half-axis of the tardy tasks, at the expense of the introduction of inequalities and variables ensuring that a takes the appropriate value[5].

For CDWP, we have to introduce a such variable a^2 to make floating the half-axis of the tardy tasks, even if $d^{\sqsupset} \geqslant p(J)$. In addition, we also have to introduce a such variable a^1 to make floating the half-axis of the early tasks, since a $d^{\sqsubset}$-straddling task can occur in all the optimal schedules, implying that the last early task do not complete at $d^{\sqsubset}$ but before. Moreover, since a non-early task is not necessarily a tardy task, two binary variables are needed for each task j: δ_j^1 indicating if j is early or not, and δ_j^2 indicating if j is tardy or not. The figure below represents the proposed encoding.



By adding for each pair of tasks i, j a variable $X_{i,j}^1$ (resp. $X_{i,j}^2$) in order to linearize $\delta_i^1\,\delta_j^1$ (resp. $\delta_i^2\,\delta_j^2$), and the four associated inequalities, one should be able to write *linear* non-overlapping inequalities managing variables $(e'_j)j \in J$ (resp. $(t'_j)j \in J$). Using e' and t' varaibles, the total penalty of the schedule can be written as follows.

$$\sum_{j \in J} \alpha_j\,(e'_j - a^1\delta_j^1) + \beta_j\,(t'j - a^2\delta_j^2)$$

Adding variables and inequalities to linearize $a\,\delta$ terms as proposed on page 70, this function can be turned into a linear function, resulting in a formulation for CDWP.

In this chapter formulations using natural variables have been introduced, and proved to be able to formulate both UCDDP and CDDP. In the next chapter, we explain how such formulations combining an exponential number of linear inequalities, integrity and extremality constraints can be solved in practice.

[5] *Cf.* Section 1.4.2, variables $(\gamma_j)_{j \in J}$ and $(b_j)_{j \in J}$, and inequalities (1.16–1.17), (1.18–1.20), (1.21), and (1.22–1.25).

Chapter 2

How to deal with non-overlapping inequalities in practice?

In the first section of this chapter, we propose a separation algorithm to handle the exponential number of non-overlapping inequalities, while in the second one, we explain how to handle both integrity and extremality constraints. This results in Branch-and-Cut algorithms for UCDDP and CDDP, which are tested and compared to other MIP formulations on the Biskup and Feldmann's benchmark in the third section.

2.1 Separation algorithm for non-overlapping inequalities

Let us briefly explain how Queyranne [34] handles the separation of inequalities (Q0), before explaining how to separate inequalities (Q1), (Q2), (Q1'), or (Q2').

2.1.1 Inequalities (Q0) separation algorithm proposed by Queyranne [34]

Let us fix $p \in {\mathbb{R}_+^*}^J$ and $C \in \mathbb{R}^J$. Deciding whether C satisfies inequalities (Q0) reduces to decide whether the maximum of the set function Γ^C is non-positive, where Γ^C is defined from parameter C as follows.

$$\forall S \subseteq J,\ \ \Gamma^C(S) = g_p(S) - p * C(S)$$

Indeed, if $\max_{S \subseteq J} \Gamma^C(S) \leqslant 0$, then C satisfies inequalities (Q0), and if conversely there exists $S \subseteq J$ such that $\Gamma^C(S) > 0$, C does not satisfy the inequality (Q0) associated with S. Using the definition of g_p, Queyranne [34] shows the following property.

Property 2.1

*Let $S \subseteq J$. **If** S maximizes Γ^C, **then** $\forall j \in J$, $j \in S \Leftrightarrow p(S) \geqslant C_j$ and $j \in J \setminus S \Leftrightarrow p(S) \leqslant C_j - p_j$.*

As a consequence of Property 2.1, a subset S maximizing Γ^C is downward closed with respect to C, *i.e.* $(C_{k'} \leqslant C_k$ and $k \in S) \Rightarrow k' \in S$, which justifies an $\mathcal{O}(n \log n)$ separation algorithm for inequalities (Q0). Indeed, once an order $\rho \in \mathcal{F}\Big([1..n], J\Big)$ such that $\Big(C_{\rho(k)}\Big)_{k \in [1..n]}$ is non-decreasing, is computed, it suffices to compute and compare the values $\Gamma^C\Big(\rho([1..k])\Big)$ for all $k \in [0..n]$, since a subset maximizing Γ^C is necessary of the form $\rho([1..k])$.

For the non-overlapping inequalities that we propose, no property similar to Property 2.1 holds, therefore this separation algorithm cannot be adapted. However, the separation problems for inequalities (Q1), (Q2), (Q1'), or (Q2') can also be solved in polynomial time using a different approach, as explained in the following.

2.1.2 Separation algorithm for inequalities (Q1), (Q2), (Q1'), or (Q2')

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'). For any parameters $(c,q)\in\mathbb{R}^J\times\mathbb{R}^{J^<}$, let us denote by $\Gamma^{c,q}$ the set function defined as follows.

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

Let $Y=(e,t,\delta,X)\in\mathbb{R}^J\times\mathbb{R}^J\times[0,1]^J\times\mathbb{R}^{J^<}$ a vector satisfying inequalities (1.5–1.8) and (X.1–X.4). The separation problem for inequalities (Q1) is to find a subset S of J such that Y does not satisfy the associated inequality (Q1) or to guarantee that Y satisfies all inequalities (Q1). It reduces to the maximization of Γ^{c^1,q^1} for the following parameters (c^1,q^1).

$$c^1=-2\left(p_j\,e_j\right)_{j\in J}\quad\text{and}\quad q^1=\left(p_i\,p_j\,(\delta_i+\delta_j-X_{i,j})\right)_{(i,j)\in J^<}$$

If $\max_{S\subseteq J}\Gamma^{c^1,q^1}(S)\leqslant 0$, then C satisfies inequalities (Q1), and if conversely there exists $S\subseteq J$ such that $\Gamma^{c^1,q^1}(S)>0$, C does not satisfy the inequality (Q1) associated with S. Indeed, we have the following equivalences.

$$\begin{aligned}Y\text{ satisfies (Q1)}&\Leftrightarrow\forall S\subseteq J,\ \sum_{(i,j)\in S^<} p_i\,p_j\,\frac{\delta_i+\delta_j-X_{i,j}}{2}\leqslant\sum_{j\in S}p_j\,e_j\\&\Leftrightarrow\forall S\subseteq J,\ \underbrace{\sum_{(i,j)\in S^<} p_i\,p_j\,(\delta_i+\delta_j-X_{i,j})-2\sum_{j\in S}p_j\,e_j}_{\Gamma^{c^1,q^1}(S)}\leqslant 0\end{aligned}$$

Similarly, the separation problem of inequalities (Q2), is equivalent to the maximization of Γ^{c^2,q^2} for the following parameters (c^2,q^2).

$$c^2=2\left((1-\delta_j)\,p_j^2-p_j\,t_j\right)_{j\in J}\quad\text{and}\quad q^2=\left(p_i\,p_j\,(2-\left(\delta_i+\delta_j\right)-X_{i,j})\right)_{(i,j)\in J^<}$$

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 ($X.3-X.4$). Therefore, let us now explain how to reduce the maximization of $\Gamma^{c,q}$ for $(c,q)\in\mathbb{R}^J\times(\mathbb{R}_+^*)^{J^<}$ to a min-cut problem in an undirected graph as proposed by Picard and Ratliff [32]. Let us assume that $J=[1..n]$ for the sake of brevity. We consider the weighted undirected graph $G=(V,A,w)$ defined as follows.

$$\begin{cases}V=[0..n+1]\\A=\{\{i,j\}\,|\,(i,j)\in V^2\text{ such that }\{i,j\}\neq\{0,n+1\}\}\\\forall(i,j)\in J^<,\ w_{\{i,j\}}=q_{i,j}\\\forall j\in J,\ w_{\{0,j\}}=[k_j]^+,\ w_{\{j,n+1\}}=[k_j]^-\quad\text{where}\quad k_j=2c_i+\sum_{i=1}^{j-1}q_{i,j}+\sum_{k=j+1}^{n}q_{j,k}\end{cases}$$

Note that V and A only depend on J, while w depends on parameters c and q. Figure 2.1 gives an illustration of such a graph for $n=5$.

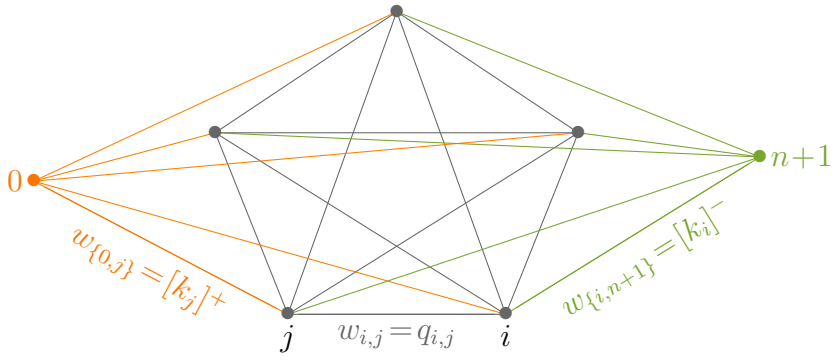


Figure 2.1: Illustration of the weighted undirected graph $G=(V,A,w)$ for $n=5$

For $(W,\overline{W})$ a bi-partition of J , let $w(W,\overline{W})$ denote the weight according to w of the corresponding **cut**, *i.e.*

$$\boldsymbol{W : \overline{W}} = \Big\{\{u,v\} \,\Big|\, u\in W, v\in\overline{W}\Big\} \quad \text{and} \quad \boldsymbol{w(W,\overline{W})} = \sum_{i,j\in W:\overline{W}} w_{\{i,j\}}.$$

In addition, let us introduce the three following quantities, which only depend on parameters (c,q).

$$Q = \sum_{(i,j)\in J^<} q_{i,j} \quad C = \sum_{j\in J} c_j \quad K = \sum_{j\in J} |k_j|$$

The following property establish, for each subset $S\subseteq J$, the link between the value of $\Gamma^{c,q}(S)$ and the weight of the cut separating s and 0 from the other nodes in G. The proof follows the line of the proof that max-cut can model [6] However, the idea to use positive and negative parts to obtain only non-negative weights on the edges is from Picard and Ratliff [32]. This proof is given on the next page.

Property 2.2 (*[32]*)

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 Gomory and Hu's algorithm [17], as explained below.

• *Implementation of the separation algorithm*

Each family of non-overlapping inequalities, *i.e.* (Q1), (Q2), (Q1'), or (Q2'), correspond to a different separation algorithm, however these algorithms are similar and consist in the same following steps.

1. Computing the weights $w_{\{i,j\}}$ according to the value of variables e,t,δ,X (resp. e',t',δ,X) in the solution provided by the solver.
2. Running the Gomory and Hu's algorithm [17] provided by the open source C++ optimization library LEMON [1], in order to obtain $\mathcal{T}$ the Gomory-Hu tree rooted in 0 .
3. Finding all minimum cost edges along the path between 0 and $n+1$ in $\mathcal{T}$.
4. For any of such edge e, testing whether the weight of the related cut is negative enough, more precisely for W the connected component of 0 in $\mathcal{T}\backslash\{e\}$ testing if $-\frac{1}{2}\, w(W,\overline{W}) + \frac{Q+C}{2} + \frac{K}{4} > 0$. If so, adding in the model the inequality (Q1) (resp. (Q2),(Q1'),(Q2') associated with $S = W\backslash\{0\}$.

Proof of Property 2.2 : The following equations show a result slightly different from the one claimed in the property. Indeed, S is a given parameter in the property while S is a minimization variable in the following computations, which allows to show how a set function *maximization* problem is turned into a *minimization* problem over cuts. However, these computations enclosed a proof of the property.

$$\begin{aligned}
\max_{S\in\mathcal{P}(J)} \Gamma^{c,q}(S) &= \max_{\delta\in\{0,1\}^J} \left(\sum_{(i,j)\in J^<} q_{i,j}\,\delta_i\,\delta_j \;+\; \sum_{i\in J} c_i\,\delta_i \right) \\
&= \max_{s\in\{-1,1\}^J} \left(\sum_{(i,j)\in J^<} q_{i,j} \left(\frac{s_i+1}{2}\right)\left(\frac{s_j+1}{2}\right) \;+\; \sum_{i\in J} c_i \left(\frac{s_i+1}{2}\right) \right) \\
&= \max_{s\in\{-1,1\}^J} \left(\sum_{(i,j)\in J^<} \frac{q_{i,j}}{4}\,[s_i\,s_j + s_i + s_j + 1] \;+\; \sum_{i\in J} \frac{c_i}{2}\,[s_i+1] \right) \\
&= \max_{s\in\{-1,1\}^J} \left(\sum_{(i,j)\in J^<} \frac{q_{i,j}}{4}\,s_i\,s_j + \sum_{i\in J} \left[\sum_{j>i} \frac{q_{i,j}}{4} + \sum_{j<i} \frac{q_{j,i}}{4} + \frac{c_i}{2} \right] s_i + \underbrace{\frac{1}{4} \sum_{(i,j)\in J^<} q_{i,j}}_{:=Q} + \underbrace{\frac{1}{2} \sum_{i\in J} c_i}_{:=C} \right) \\
&= \frac{1}{4} \max_{s\in\{-1,1\}^J} \left(\sum_{(i,j)\in J^<} \underbrace{q_{i,j}}_{:=w_{i,j}}\,s_i\,s_j \;+\; \sum_{i\in J} \underbrace{\left[2\,c_i + \sum_{j>i} q_{i,j} + \sum_{j<i} q_{j,i} \right]}_{:=\alpha_i} s_i \right) + \frac{Q}{4} + \frac{C}{2} \\
&= \frac{1}{4} \max_{s\in\{-1,1\}^J} \left(\sum_{(i,j)\in J^<} w_{i,j}\,s_i\,s_j \;+\; \sum_{i\in J} [w_{0,i} - w_{i,n+1}]\,s_i \right) + \frac{Q}{4} + \frac{C}{2} \text{ since } \begin{cases} w_{0,i} = [\alpha_i]^+ \\ w_{i,n+1} = [\alpha_i]^- \end{cases} \\
&= \frac{1}{4} \max_{\substack{s_0=1 \\ s\in\{-1,1\}^J \\ s_{n+1}=-1}} \left(\sum_{(i,j)\in J^<} w_{i,j} \underbrace{s_i\,s_j}_{\substack{\uparrow \\ =1 \text{ if } s_i=s_j \\ =-1 \text{ otherwise}}} + \sum_{i\in J} w_{0,i} \underbrace{s_0\,s_i}_{\substack{\uparrow \\ =1 \text{ if } s_i=s_0 \\ =-1 \text{ otherwise}}} + \sum_{i\in J} w_{i,n+1} \underbrace{s_i\,s_{n+1}}_{\substack{\uparrow \\ =1 \text{ if } s_i=s_{n+1} \\ =-1 \text{ otherwise}}} \right) + \frac{Q}{4} + \frac{C}{2} \\
&= \frac{1}{4} \max_{\substack{s_0=1 \\ s\in\{-1,1\}^J \\ s_{n+1}=-1}} \left(\sum_{(i,j)\in J^<} w_{i,j} - 2 \sum_{\substack{(i,j)\in J^< \\ s_i\neq s_j}} w_{i,j} + \sum_{i\in J} w_{0,i} - 2 \sum_{\substack{i\in J \\ s_i\neq s_0}} w_{0,i} + \sum_{i\in J} w_{i,n+1} - 2 \sum_{\substack{i\in J \\ s_i\neq s_{n+1}}} w_{i,n+1} \right) + \frac{Q}{4} + \frac{C}{2} \\
&= \frac{-2}{4} \min_{\substack{s_0=1 \\ s\in\{-1,1\}^J \\ s_{n+1}=-1}} \left(\sum_{\substack{(i,j)\in V^< \\ s_i\neq s_j}} w_{i,j} \right) \;+\; \underbrace{\frac{1}{4} \sum_{(i,j)\in J^<} w_{i,j}}_{=Q} \;+\; \frac{1}{4} \sum_{i\in J} \underbrace{[w_{0,i} + w_{i,n+1}]}_{=[\alpha_i]^+ + [\alpha_i]^-} + \frac{Q}{4} + \frac{C}{2} \\
&= -\frac{1}{2} \min_{\substack{s_0=1 \\ s\in\{-1,1\}^J \\ s_{n+1}=-1}} \left(\sum_{\substack{(i,j)\in V^< \\ s_i\neq s_j}} w_{i,j} \right) \;+\; \frac{Q}{4} \;+\; \underbrace{\frac{1}{4} \sum_{i\in J} \left| 2c_i + \sum_{j>i} q_{i,j} + \sum_{j<i} q_{j,i} \right|}_{=K} \;+\; \frac{Q}{4} + \frac{C}{2} \\
&= -\frac{1}{2} \left(\begin{array}{l} \text{minimum weight of a cut} \\ \text{in the complete undirected graph} \\ \text{over the vertex set } J \cup \{0, n+1\} \\ \text{for the edge weight } w \end{array} \right) \;+\; \frac{Q}{2} + \frac{C}{2} + \frac{K}{4}
\end{aligned}$$

□

2.2 Extremality and integrality constraints

The aim of this section is to show that Formulations F^3 and F^4 can be solved by a classical Branch-and-Cut algorithm, despite they combine extremality and integrity constraints. Let us show this result only for Formulation F^3, since the arguments used can easily be extended to Formulation F^4. Let us consider the three following relaxations of F^3.

$$F^3\text{-LP}: \min_{(e,t)\in\pi_{e,t}(P^3)} g_{\alpha,\beta}(e,t)$$

$$F^3\text{-EXTR}: \min_{(e,t)\in\pi_{e,t}\left(\operatorname{extr} P^3\right)} g_{\alpha,\beta}(e,t)$$

$$F^3\text{-INT}: \min_{(e,t)\in\pi_{e,t}\left(\operatorname{int}_\delta P^3\right)} g_{\alpha,\beta}(e,t)$$

The formulation F^3-LP is obtained by relaxing both integrity and extremality conditions. It is a linear program defined by an exponential number of inequalities. As explain in the previous section, the separation problem associated with the non-overlapping inequalities defining P^3 is solvable in polynomial time. Therefore, F^3-LP can be solved in polynomial time using a cutting plane algorithm [19].

Moreover, using the simplex algorithm to solve each LP-relaxation of the cutting plane algorithm, the extremality of the global solution is ensured. Indeed, such a solution is then an extreme point of the last polyhedron P considered by the algorithm, which necessarily contains P^3 (*Cf.* Lemma CA.32). Then in this case, solving F^3-LP is equivalent to solving F^3-EXTR.

A classical way to manage the integrity constraint is to use a Branch-and-Bound algorithm, and even a Branch-and-Cut algorithm when an exponential number of inequalities has to be managed. That means that each node of the Branch-and-Bound algorithm consists in a cutting plane algorithm. In general, even if the algorithm used to solve each linear relaxation provides an extreme point, the final solution provided by the Branch-and-Cut algorithm is not an extreme point of the initial polyhedron. Example 5 illustrates such a case. However, since the integrity constraint in F^3 applies only on boolean variables, the problem cannot appear, as stated in the following property.

Property 2.3

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 F^3-INT by branching on δ variables solves F^3.

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^3.
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 $P^3 \cap F^{J_0,J_1}$ 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 $Y = (e,t,\delta,X) \in \operatorname{extr}\left(P^3 \cap F^{J_0,J_1}\right)$.
By definition of $E(\delta)$ and $T(\delta)$, $Y \in P^3 \cap F^{T(\delta),E(\delta)}$. Moreover, $J_1 \subseteq E(\delta)$ and $J_0 \subseteq T(\delta)$, thus we have $P^3 \cap F^{T(\delta),E(\delta)} \subseteq P^3 \cap F^{J_0,J_1}$. Using Lemma CA.32, we deduce that $Y \in \operatorname{extr}\left(P^3 \cap F^{T(\delta),E(\delta)}\right)$.
Since $P^3 \cap F^{T(\delta),E(\delta)}$ is exactly the set denoted by P^δ in the proof of Theorem 1.14, we can similarly show that $\operatorname{extr}(P^3 \cap F^{T(\delta),E(\delta)}) \subseteq \operatorname{extr}(P^3)$. We deduce that $Y \in \operatorname{extr}(P^3)$. □

Example 5 : *A formulation combining extremality and integrity constraints which cannot be solved by a standard Branch-and-Bound algorithm*

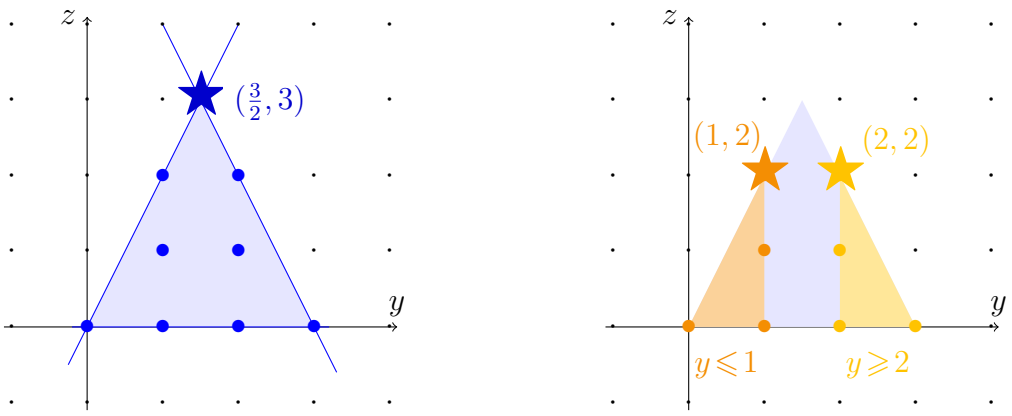
Let us consider the following formulation which presents both integrity and extremality constraints, where int_y denotes the operator keeping only the points (y, z) such that y is an integer.

$$F : \max_{(y,z)\in \text{int}_y(\text{extr}\, P)} z \quad \text{where } P = \left\{(y,z)\in\mathbb{R}_+\times\mathbb{R}_+ \mid \; z\leqslant 2\,y, \;\; z\leqslant -2y+7\right\}$$

Let us consider a Branch-and-Bound algorithm $\mathcal{A}$, where the LP-relaxation at each node provides an extreme point. For formulation F, $\mathcal{A}$ provides a solution which does not belong to $\text{extr}(P)$, and a value which is larger than the true optimum.

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 $(1, 2)$, and $(2, 2)$. Since they are integer, the Branch-and-Bound stops at these depth, and the solution returned is one of these two points, which are not extreme points of P. Moreover, the returned optimal value is 2, whereas the true optimal value is 0, since P admits only two integer extreme points, $(0, 0)$ and $(3, 0)$.

The figure below represents the polyhedra considered at each node of the algorithm $\mathcal{A}$, and for each one the extreme optimal point obtained is represented with a star. Dots represent integer points, and bullet points the integer points in each polyhedron.



Integer points and maximizer at the root node (left) or for nodes of depth 1 (right)

Remark 2.4

Note that we have already shown, in the proof of Theorem 1.12, that F^3 and F^3-INT have the same optimal value, *i.e.* $\text{val}(F^3) = \text{val}(F^3\text{-INT})$. However, this observation on the optimal *values* is not sufficient when a feasible *solution* for UCDDP has to be found. Therefore, we have explained in this section, how to solve F^3-INT to obtain a solution of F^3, that is a vector encoding an optimal schedule.

Using the results of these last two sections, we have implemented Branch-and-Cut algorithms for Formulations F^3 and F^4, using the MIP solver CPLEX. *The next section presents and compares their performances.*

2.3 Experimental results

In order to assess the efficiency of Formulations F^2, F^3, and F^4, we compare them to other MIP formulations proposed in the literature. Therefore, we have implemented Formulations F_{LO} and F_{TI} presented in Section 0.4.

All the experiments presented in this section 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 MIP solver CPLEX version 12.6.3.0. The branching scheme and the management of the current bounds are done by CPLEX. Note that only δ variables are set as integer variables for F^2, F^3, and F^4, hence the branching decision only involves δ, not X. For Formulations F^3, and F^4, the separation of inequalities (Q1) and (Q2) (resp. (Q1') and (Q2')) is implemented within the so-called Callback functions proposed by CPLEX, using the open source C++ optimization library LEMON [1] (*Cf.* Section 2.1). For the the sake of comparison, all the formulations use CPLEX Default. Moreover, the time limit is set to 3600 seconds.

2.3.1 Benchmarks used to compare formulations

• *Biskup and Feldmann's benchmark*

We test the different formulations on the benchmark proposed by [9], available online on OR-Library [8]. 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.

Sourd [41] 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 $\mathcal{O}(n\, p(J))$ 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 task processing times, which ensures a fast computation time of the Lagrangian lower bound.

• *A new benchmark with long processing times*

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 $p_{max} \in \{100, 200, 300\}$. We talk about "long processing times" by comparison with processing times in the Biskup and Feldmann's benchmark, whose ranges are $[1, 20]$.

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.

• *Entries of the following tables*

n:	the number of tasks
h:	the parameter setting the due date d to $\lfloor h\,p(J) \rfloor$ (for CDDP only)
#opt:	number of instances optimally solved under the 3600 seconds time limit (among the 10 proposed by the benchmark)
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

2.3.2 Formulations for UCDDP

In this section, we compare Formulations F^3, F^2,F_{LO} and F_{TI} for UCDDP instances. Table 2.1 presents the results obtained on Biskup and Feldmann's benchmark, while Table 2.2 presents those obtained on long processing times instances, for $p_{max} \in \{100, 200, 300\}$.

n	F_{LO}			F_{TI}			F^3			F^2		
	#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 2.1: Solving Biskup and Feldmann's unrestrictive instances using F_{LO}, F_{TI}, F^3 and F^2

As shown in Table 2.1, 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. F^3 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 F^3 fails to solve 50-task instances within the time limit. However, other experiments show that, under a time limit of 10 000 seconds, F^3 solves 9 over 10 instance for $n = 50$, with an average computation time of 4721 seconds. F^2 is able to optimally solve all instances up to size 50 within the time limit. Other experiments conducted without CPLEX features show that F^2 can be faster: 22 seconds for $n = 40$, 215 seconds for $n = 50$ and 4063 seconds for $n = 60$.

As shown in Table 2.2, 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 F_{TI} only takes 40 seconds in average to solve all the 50-task Biskup and Feldmann's instances, it only solves 8 over 10 instances for $n = 50$ 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 $p_{max} = 300$. Conversely, F^3 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 F^2 up to size 40 regardless of p_{max} value.

p_{max}	n	F_{TI}			F^3			F^2		
		#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 2.2: Solving unrestrictive instances generated with p_{max} 100, 200, 300 using F_{TI}, F^3 and F^2

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 F^3 and F^2 do not significantly get worse with processing time increase.

2.3.3 Formulations for CDDP

In this section, we compare Formulations F^4, F_{LO} and F_{TI} for CDDP instances. Table 2.3 presents the results obtained on the Biskup and Feldmann's benchmark, while Table 2.4 presents those obtained on long processing times instances, for $p_{max} = 200$.

As shown in Table 2.3, F_{LO} is unable to solve any CDDP 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} can solve no 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 CDDP instances. Moreover, the computation times are similar to those obtained for the UCDDP instances: less than 22 seconds for 30-task instances.

F^4 is able to optimally solve all the instances up to size 20 along with 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 $n = 30$ and $h = 0.2$.

n	h	F_{LO}			F_{TI}			F^4		
		#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 2.3: Solving Biskup and Feldmann's restrictive instances using F_{LO}, F_{TI} and F^4

As shown in Table 2.4, for long processing time instances with $p_{max} = 200$, F_{TI} optimally solves almost all the instances within the time limit up to size 20. It is important to notice that, for similar sizes, F_{TI} computation time is significantly larger for long processing time instances than for Biskup and Feldmann's ones: for $n = 20$, at least 116 seconds against a few seconds. In contrast, F^4 optimally solves instances up to size 20, along with 30-task instances when $h = 0.2$. Note that, for the long processing time instances considered here, F^4 is rather faster than F_{TI} up to size 20. However, F^4 fails to solve instances with $n = 30$.

p_{max}	n	h	F_{TI}			F^4		
			#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 2.4: Solving restrictive instances generated with $p_{max} = 200$ using F_{TI} and F^4

For CDDP instances, we obtain the same conclusion as the one drawn for UCDDP instances. F_{TI} is faster than F^4 for Biskup and Feldmann's instances, while this is not the case for long processing times instances.

Other experiments show that F^4 used on unrestrictive Biskup and Feldmann's instances (*i.e.* with $h=1$) is less efficient than F^3: 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 (*Cf.* page 12) an *unrestrictive* due date as a due date larger than the total length of tasks, *i.e.* $d \geqslant p(J)$. It is a common definition. However, according to Biskup and Feldmann [9], a due date d must be said unrestrictive if solving the problem for an arbitrarily large due date d gives a solution which can be transposed for d (*Cf.* page 24).

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. Therefore, this definition depends on the algorithm chosen to solve, and even sometimes on the execution of this algorithm. Second, 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. However, considering the point of view of the second definition leads to the following **F^2-F^4 procedure** to solve a CDDP instance.

1. Solving the instance as an UCDDP instance (*i.e.* without considering d) using Formulation F^2.
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 as a CDDP instance (*i.e.* by considering d) using F^4.

For the Biskup and Feldmann's benchmark, the total length of the early tasks in the optimal solutions is, on average, 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 in [9]. For these instances, the F^2-F^4 procedure can be relevant.

As shown by the previous experiments, Formulation F^2 is more promising than Formulation F^3 and F^4. Furthermore, other experiments show that F^2 relaxation value is very low (the gap to the optimal value is about 90% for $n=60$, Cf. Section 6.2), which let us think that there is a room of improvement for F^2 regarding its linear relaxation value.
Therefore, we focus in the next parts on the reinforcement of Formulation F^2. Another reason to focus on this formulation, is that valid inequalities for F^2 are also valid inequalities for F^3 and F^4, since variables δ, X are used as partition variables in the three formulations. Actually, this kind of partition variables are also used to formulate the MAX-CUT *problem, as proposed by Padberg [31]. The next chapter aims at transpose known results for the* MAX-CUT *formulations to F^2.*

PART B

Reinforcement inequalities for δ, X variables

In Section 0.5, we have introduced δ, X variables to encode the ordered bi-partitions of J, in order to reformulate UCDDP as a MIP. This way, we proposed Formulation F^2. Subsequently, δ, X variables were also used in F^3 and F^4. In these three formulations, the consistency between δ and X values was dealt using only inequalities ($X.1$–$X.4$), since they are sufficient when δ is $\{0,1\}$-valued.

However, when δ is no longer subject to an integrity constraint, these inequalities appear to be poor, in the sense that the optimal value for F^2-LP, the linear relaxation of F^2, is much lower than the one for F^2. Indeed, the underlying polyhedron $P^n_{F^2}$ admits fractional extreme point — typically points where some δ variables take the value $1/2$ — leading to an artificially low value according to the $h_{\alpha,\beta}$). Therefore, it is interesting to study $P^n_{\delta,X}$, the convex hull of integer points (δ, X) that encode an ordered bi-partition of $[1..n]$. Any valid inequality for $P^n_{\delta,X}$ can be used to reinforce Formulations F^2, F^3, and F^4.

In Chapter 3, we present valid inequalities for $P^n_{\delta,X}$ deduced from the literature for two related polytopes: the cut polytope CUT^n and the Boolean quadric polytope QP^n. Indeed, if $(\delta, X) \in P^n_{\delta,X}$ encodes the ordered bi-partition (E, T) of $[1..n]$, then the set $\{\{i, j\} \mid (i, j) \in [1..n]^<, X_{i,j} = 1\}$ is the cut induced by the partition $\{E, T\}$ in the complete graph on $[1..n]$, consequently $X \in CUT^n$. However, note that projecting on X variables causes a loss of information: while (δ, X) encodes an ordered bi-partition (E, T), X only encodes a partition $\{E, T\} = \{T, E\}$, where E and T are interchangeable. Conversely, points in QP^n also encode ordered bi-partitions.

Even if Chapter 3 is mainly a state of the art about facet inequalities of CUT^n and QP^n, we also propose a framework to transpose inequalities from a polytope to another. We focus on transpositions for CUT^n, QP^n and $P^n_{\delta,X}$, since our aim is to reinforce Formulation F^2. We present experimental results that show an improvement in the linear relaxation value, but also an increase in the solving time when reinforcement inequalities are added.

In Chapter 4, we propose another approach to reinforce F^2, by taking into account the objective function, and not only the structure of $P^n_{\delta,X}$. The idea — which also leads Part C — is to cut off the integer points that have an high value according to the objective function instead of cutting off fractional points. Chapter 4 is a first application of this idea: we provide inequalities that cut off the integer point 0, since it encodes a solution having a too high value. More precisely, we provide *facet defining* inequalities of the polytope of non-trivial cuts encoded by δ, X variables in a complete graph.

> *NB*: In Chapters 3 and 4, the scheduling notations $(J, n, d, \alpha, \beta, p, E, T, f, \dots)$ no longer apply, and these symbols may even be used for different purposes.

Chapter 3

Bridging polytopes CUT^n, QP^n and $P^n_{\delta,X}$

In Section 3.1, we present the cut polytope and the Boolean quadric polytope, *i.e.* CUT^n and QP^n, before giving, in Section 3.2, some valid and facet defining inequalities for each one. In Section 3.3, we give further results about the inequalities mentioned for QP^n. These results are established by Padberg in [31], however we provide a new explicit proof for some of them. In Section 3.4, we give a formal framework to transpose valid and facet defining inequalities from a polyhedron to a similar one in general. This framework can be used between a polyhedron and its projection, between two polyhedra that are equivalent after a change of variables, or even between a polyhedron and itself when it presents some symmetries. At the end of the section, we use it in these three ways for CUT^n, QP^n and $P^n_{\delta,X}$. Our aim is to use the facet defining inequalities provided in the literature, to reinforce Formulation F^2 in Section 3.5. Some experimental results are reported.

Before introducing CUT^n and QP^n, let us introduce some notations.

For two given disjoints sets A and B, let us denote by $A{:}B$ the set of all the pair sets that have one element in A and the other in B, *i.e.* $\boldsymbol{A:B} = \{\{u,v\} \,|\, u \in A, v \in B\}$.
As previously, if a vector a is indexed by a set S, for any subset $T \subseteq S$, $a(T)$ is the sum of the a components whose index is in T, *i.e.* $\boldsymbol{a(T)} = \sum_{i \in T} a_i$.
In particular, for two disjoint sets A and B, $\boldsymbol{a(A:B)} = \sum_{u \in A} \sum_{v \in B} a_{\{u,v\}}$.

In part B, we consider only undirected graphs. Then, for any node set $V \subseteq \mathbb{N}$, there is a one-to-one correspondence between the edge set E and the pair set $V^<$ defined as follows.

$$E = \left\{ \{u,v\} \,\middle|\, (u,v) \in V^2 \right\} \quad \text{and} \quad V^< = \left\{ (u,v) \,\middle|\, (u,v) \in V^2 \text{ s.t. } u < v \right\}$$

Following the same line, we introduce the following set of 3-tuples to represent the set of 3-node subsets.

$$\boldsymbol{V^{<<}} = \left\{ (i,j,k) \in V^3 \,\middle|\, i < j < k \right\}$$

We use such tuple sets to describe inequality families. For example, we will write the two inequality families $\begin{cases} \forall (i,j) \in V^<,\ X_{i,j} \leqslant \delta_i \\ \forall (i,j) \in V^<,\ X_{i,j} \leqslant \delta_j \end{cases}$ instead of the single inequality family $\forall \{i,j\} \in V^<,\ X_{\{i,j\}} \leqslant \delta_i$.
However, for sake of brevity, if a is a vector indexed by $V^<$ and $(u,v) \in V^2$ an arbitrary pair of nodes, we simply denote by $a_{u,v}$ the component of a indexed by $(\min(u,v), \max(u,v))$.

Finally, for any graph G, let us denote by $\boldsymbol{\mathcal{C}y}(G)$ the set of elementary cycles of G.

3.1 Polytopes CUT^n, QP^n and $P^n_{\delta,X}$

Let us fix $n \in \mathbb{N}^*$. We denote by $K_n = (V, E)$ the complete undirected graph on $[1..n]$, *i.e.* $V = [1..n]$ and E can be identified to $V^<$. Moreover, we denote by m the number of edges, *i.e.* $m = |E| = \frac{n\,(n-1)}{2}$.

• *CUT^n definition*

For any $S \subseteq V$, we define $\chi^S \in \mathbb{R}^E$ as $\boldsymbol{\chi^S} = \mathbb{I}_{S:V\setminus S}$.

The **cut polytope** CUT^n is then defined as follows.

$$\boldsymbol{CUT^n} = \text{conv}\,(S^n_X) \quad \text{where} \quad \boldsymbol{S^n_X} = \left\{ \chi^S \,\middle|\, S \subseteq V \right\}$$

• *QP^n definition*

For any $S \subseteq V$, we define $z^S \in \mathbb{R}^V \times \mathbb{R}^E$ as $\boldsymbol{z^S} = (\mathbb{I}_S, \mathbb{I}_{S^<})$.

The following lemma is the equivalent of Lemma 0.5 for new binary variables x, y.

Lemma 3.1

Let $z = (x, y) \in \mathbb{R}^V \times \mathbb{R}^E$.

***If** $x \in \{0,1\}^V$ and (x, y) satisfies the following inequalities, called **trivial inequalities**:*

$$\forall (i,j) \in V^<,\ \ y_{i,j} \geqslant (x_i + x_j) - 1 \tag{3.1}$$
$$\forall (i,j) \in V^<,\ \ y_{i,j} \leqslant x_i \tag{3.2}$$
$$\forall (i,j) \in V^<,\ \ y_{i,j} \leqslant x_j \tag{3.3}$$
$$\forall (i,j) \in V^<,\ \ y_{i,j} \geqslant 0 \tag{3.4}$$

then $\forall (i,j) \in J^<,\ y_{i,j} = \begin{cases} 1 \text{ if } x_i = 1 \text{ and } x_j = 1 \\ 0 \text{ otherwise} \end{cases}$, *therefore $z = z^S$ for $S = \{u \in V \mid x_u = 1\}$.*

***Conversely**, for any $S \subseteq V$, z^S satisfies inequalities (3.1–3.4).*

Note that x variables are equivalent to δ variables, but y variables identify the pair of nodes having the same value 1 according to x, while X identifies the pair of nodes having different values according to δ. Thanks to this lemma, we can define the set $S^n_{x,y}$ in a twofold manner as follows.

$$\boldsymbol{S^n_{x,y}} = \left\{ z^S \,\middle|\, S \subseteq V \right\} = \left\{ (x,y) \in \{0,1\}^V \times \{0,1\}^E \,\middle|\, (x,y) \text{ satisfies } (3.1-3.4) \right\}$$

The **boolean quadric polytope** QP^n and its linear relaxation QP^n_{LP} are then defined as follows.

$$\boldsymbol{QP^n} = \text{conv}\left(S^n_{x,y}\right) \quad \text{and} \quad \boldsymbol{QP^n_{\text{LP}}} = \left\{ (x,y) \in \mathbb{R}^V \times \mathbb{R}^E \,\middle|\, (x,y) \text{ satisfies } (3.1-3.4) \right\}$$

Remark 3.2

Using points z^S where S is a singleton or a pair, it can be shown that $\dim(S^n_{x,y}) = \dim(\mathbb{R}^V \times \mathbb{R}^E) = \frac{n\,(n+1)}{2}$.
Moreover, it can be shown that points z^S are extreme points in QP^n_{LP}, consequently $S^n_{x,y} = \text{extr}(QP^n)$

• *$P^n_{\delta,X}$ definition*

In order to have homogeneous notations in this chapter, we introduce the following notations.

$$\boldsymbol{P^n_{\delta,X}} = \text{conv}\left(S^n_{\delta,X}\right) \quad \text{where} \quad \boldsymbol{S^n_{\delta,X}} = \left\{ (\delta, X) \in \{0,1\}^V \times \{0,1\}^E \,\middle|\, \forall (i,j) \in E,\ X_{i,j} = 1 \Leftrightarrow \delta_i \neq \delta_j \right\}$$

According to Lemma 0.5, the set $S^n_{\delta,X}$ is exactly the set of integer points in $P^n_{F^2}$, *i.e.* $S^n_{\delta,X} = \text{int}_\delta(P^n_{F^2})$.

3.2 Classical inequalities for QP^n and CUT^n

In this section, we just give some classical inequalities for QP^n and CUT^n, presented in [7] and [31].

- *Triangle inequalities for x, y variables*

$$\forall (i,j,k) \in V^{<<},\ \ x_i + x_j + x_k \leqslant 1 + y_{i,j} + y_{i,k} + y_{j,k} \tag{3.5}$$
$$\forall (i,j,k) \in V^{<<},\ \ + y_{i,j} + y_{i,k} - y_{j,k} \leqslant x_i \tag{3.6}$$
$$\forall (i,j,k) \in V^{<<},\ \ + y_{i,j} - y_{i,k} + y_{j,k} \leqslant x_j \tag{3.7}$$
$$\forall (i,j,k) \in V^{<<},\ \ - y_{i,j} + y_{i,k} + y_{j,k} \leqslant x_k \tag{3.8}$$

- *Clique inequalities for x, y variables*

$$\forall S \subseteq V,\ \forall \theta \in \Big[1\,..\,|S|-2\Big],\ \ \theta\, x(S) - y(S^<) \leqslant \frac{\theta\,(\theta+1)}{2} \tag{3.9}$$

- *Cut inequalities for x, y variables*

$$\forall (S,T) \subseteq \mathcal{P}(V)^2 \text{ s.t. } S \cap T \neq \emptyset, |S| \geqslant 1, |T| \geqslant 2,\ \ -x(S) - y(S^<) - y(T^<) + y(S:T) \leqslant 0 \tag{3.10}$$

- *Generalized cut inequalities for x, y variables*

$$\begin{array}{l} \forall (S,T) \subseteq \mathcal{P}(V)^2 \text{ s.t. } S \cap T \neq \emptyset,\ s = |S| \geqslant 1,\ t = |T| \geqslant 2, \\ \qquad (s-t)\, x(S) + (t-s-1)\, x(T) - y(S^<) - y(T^<) + y(S:T) \leqslant \dfrac{(t-s)\,(t-s-1)}{2} \end{array} \tag{3.11}$$

- *Odd cycle inequalities for x, y variables*

$$\forall C \subseteq \mathcal{C}y(K_n),\ \forall F \subseteq C \text{ s.t. } |F| \text{ is odd},\ \ x(S^0) - x(S^2) + y(C \setminus F) - y(F) \leqslant \left\lfloor \frac{|F|}{2} \right\rfloor \tag{3.12}$$
$$\text{where } \begin{cases} S^0 = \{u \in \text{supp}(C) \,|\, u \text{ is adjacent to 0 edges in } C \setminus F\} \\ S^2 = \{u \in \text{supp}(C) \,|\, u \text{ is adjacent to 2 edges in } C \setminus F\} \end{cases}$$

Property 3.3 (*Padberg [31]*)

Inequalities (3.5 – 3.8), (3.9), (3.10), and (3.11) define facets of QP^n.

- *Triangle inequalities for X variables*

$$\forall (i,j,k) \in V^{<<},\ \ + X_{i,j} - X_{i,k} - X_{j,k} \leqslant 0 \tag{3.13}$$
$$\forall (i,j,k) \in V^{<<},\ \ - X_{i,j} + X_{i,k} - X_{j,k} \leqslant 0 \tag{3.14}$$
$$\forall (i,j,k) \in V^{<<},\ \ - X_{i,j} - X_{i,k} + X_{j,k} \leqslant 0 \tag{3.15}$$
$$\forall (i,j,k) \in V^{<<},\ \ + X_{i,j} + X_{i,k} + X_{j,k} \leqslant 2 \tag{3.16}$$

- *Odd sub-cycle inequalities for X variables*

$$\forall C \subseteq \mathcal{C}y(K_n),\ \forall F \subseteq C \text{ s.t. } |F| \text{ is odd},\ \ X(F) - X(C \setminus F) \leqslant |F| - 1 \tag{3.17}$$

- *Clique inequalities for X variables*

$$\forall S \subseteq V,\ X(S^<) \leqslant \left\lfloor \frac{|S|}{2} \right\rfloor \left\lceil \frac{|S|}{2} \right\rceil \tag{3.18}$$

Property 3.4 (*Barahona and Mahjoub [7]*)

Inequalities (3.13 – 3.16) and (3.18) associated with a set S such that $|S|$ odd, define facets of CUT^n.

3.3 Some results about QP^n_{LP}

We present in this section some results about QP^n_{LP} given in [31] that enlighten the link between trivial, triangle and odd sub-cycle inequalities. More precisely Property 3.8 states that odd sub-cycle inequalities are redundant with triangle inequalities, and Corollary 3.9 specifies in which manner triangle inequalities reinforce trivial inequalities.

Corollary 3.9 relies on Properties 3.5 and 3.8, which are proved in [31], but also on the fact that an extreme point of QP^n_{LP} violates at least one odd sub-cycle inequality. Since we did not find the proof of this fact in [31], we propose to prove it in two steps, namely Lemmas 3.6 and 3.7.

Property 3.5 (*Padberg [31]*)

The extreme points of QP^n_{LP} are $\{0, 1, \frac{1}{2}\}$-valued, i.e. $\forall z \in \text{extr}(QP^n_{LP})$, $z \in \{0, 1, \frac{1}{2}\}^V \times \{0, 1, \frac{1}{2}\}^E$.

Lemma 3.6

Let $z = (x, y) \in \mathbb{R}^V \times \mathbb{R}^E$. Let $S = \left\{u \in V \,\middle|\, x_u = \frac{1}{2}\right\}$.
***If** $z \in \text{extr}\,(QP^n_{LP})$, **then** either $|S| = 0$ and $z \in S^n_{x,y}$, or $|S| \geqslant 3$.*

Proof: Let us introduce additional notations: $S^0 = \{i \in V \mid x_i = 0\}$ and $S^1 = \{i \in V \mid x_i = 1\}$.
Then, according to Property 3.5, the node (resp. edge) set of K_n can be decomposed as follows.

$$V = S \sqcup S^0 \sqcup S^1 \quad \text{and} \quad E = S^< \sqcup (S:S^0) \sqcup (S:S^1) \sqcup (S^0)^< \sqcup (S^0:S^1) \sqcup (S^1)^<$$

Thanks to Lemma 3.1, we have $y_{i,j} = 1$ for any $(i,j) \in (S^1)^<$ and $y_{i,j} = 0$ for any $(i,j) \in S^0{:}S^1 \sqcup (S^0)^<$.
Moreover, for $(i,j) \in S:S^0$, we have $y_{i,j} \leqslant x_j = 0$ from inequality (3.3) and $y_{i,j} \geqslant 0$ from (3.4), thus $y_{i,j} = 0$.
For any $(i,j) \in S:S^1$, we have $y_{i,j} \leqslant x_i = \frac{1}{2}$ from (3.2) and $y_{i,j} \geqslant (x_i + x_j) - 1 = \frac{1}{2} + 1 - 1$ from (3.1), thus $y_{i,j} = \frac{1}{2}$.

For $|S| \in \{1, 2\}$, the idea is to exhibit two points of $S^n_{x,y}$ whose middle is z, which contradicts the extremality of z. We build these points from z by only changing its $\frac{1}{2}$-valued components.

• Let us assume that $|S| = 1$. Let us denote by i_0 the single element of S, *i.e.* $S = \{i_0\}$.
We define two points $z^1 = (x^1, y^1)$ and $z^0 = (x^0, y^0)$ as follows.

$$x^1_{i_0} = 1, \quad \forall i \in V \setminus S,\ x^1_i = x_i, \quad \forall j \in S^1,\ y^1_{i_0,j} = 1, \quad \forall (i,j) \in E \setminus (S:S^1),\ y^1_{i,j} = y_{i,j}$$

$$x^0_{i_0} = 0, \quad \forall i \in V \setminus S,\ x^0_i = x_i, \quad \forall j \in S^1,\ y^0_{i_0,j} = 0, \quad \forall (i,j) \in E \setminus (S:S^1),\ y^0_{i,j} = y_{i,j}$$

Using the tables on the right that sum up the values of z, z^1, and z^0, one can easily check that $z = \frac{z^1 + z^0}{2}$.
Moreover, using Lemma 3.1, one can also check that $z^0 \in S^n_{x,y} \subseteq QP^n_{\text{LP}}$ and $z^1 \in S^n_{x,y} \subseteq QP^n_{\text{LP}}$.
Then z is the middle of two other points in QP^n_{LP}. A contradiction.

i	x_i	x^1_i	x^0_i
S	1/2	1	0
S^0	0	0	0
S^1	1	1	1

e	y_e	y^1_e	y^0_e
$S:S^0$	0	0	0
$S:S^1$	1/2	1	0
$(S^0)^<$	0	0	0
$S^0:S^1$	0	0	0
$(S^1)^<$	1	1	1

• Let us assume that $|S| = 2$. Let us denote by i_0 and j_0 the two elements of S, *i.e.* $S = \{i_0, j_0\}$.
We have $y_{i_0,j_0} \leqslant x_{i_0} = \frac{1}{2}$ from (3.2) and $y_{i_0,j_0} \geqslant (x_{i_0} + x_{j_0}) - 1 = \frac{1}{2} + \frac{1}{2} - 1$ from (3.1), thus $y_{i_0,j_0} \in [0, \frac{1}{2}]$.
Since z is an extreme point, necessarily $y_{i_0,j_0} \in \{0, \frac{1}{2}\}$, otherwise for $\varepsilon \in \mathbb{R}^*_+$ small enough, $z + \varepsilon\, \mathbb{I}_{i_0,j_0}$ and $z + \varepsilon\, \mathbb{I}_{i_0,j_0}$ would satisfy (3.1 – 3.4) and z would be the middle of two other points in QP^n_{LP}.
Therefore, we have two cases to consider.

– If $y_{i_0,j_0}=0$, then we define two points $z^1=(x^1,y^1)$ and $z^2=(x^2,y^2)$ as follows.

$$x^1_{i_0}=1,\ x^1_{j_0}=0,\quad \forall i\in V\setminus S,\ x^1_i=x_i,\qquad \forall j\in S^1,\ y^1_{i_0,j}=1,\ y^1_{j_0,j}=0,\quad \forall (i,j)\in E\setminus(S:S^1),\ y^1_{i,j}=y_{i,j}$$

$$x^2_{i_0}=0,\ x^2_{j_0}=1,\quad \forall i\in V\setminus S,\ x^2_i=x_i,\qquad \forall j\in S^1,\ y^2_{i_0,j}=0,\ y^2_{j_0,j}=1,\quad \forall (i,j)\in E\setminus(S:S^1),\ y^2_{i,j}=y_{i,j}$$

Using the tables on the right that sum up the values of z, z^1, and z^2, one can easily check that $z=\frac{z^1+z^2}{2}$.

Moreover, using Lemma 3.1, one can also check that $z^1\in S^n_{x,y}\subseteq QP^n_{\mathrm{LP}}$ and $z^2\in S^n_{x,y}\subseteq QP^n_{\mathrm{LP}}$.

Then z is the middle of two other points in QP^n_{LP}. A contradiction.

i	x_i	x^1_i	x^2_i
$\{i_0\}$	1/2	1	0
$\{j_0\}$	1/2	0	1
S^0	0	0	0
S^1	1	1	1

e	y_e	y^1_e	y^2_e
$\{i_0,j_0\}$	0	0	0
$\{i_0\}:S^0$	0	0	0
$\{i_0\}:S^1$	1/2	1	0
$\{j_0\}:S^0$	0	0	0
$\{j_0\}:S^1$	1/2	0	1
$(S^0)^<$	0	0	0
$S^0:S^1$	0	0	0
$(S^1)^<$	1	1	1

– If $y_{i_0,j_0}=\frac{1}{2}$, then we define two points $z^1=(x^1,y^1)$ and $z^0=(x^0,y^0)$ as follows.

$$x^1_{i_0}=1,\ x^1_{j_0}=1,\ \forall i\in V\setminus S,\ x^1_i=x_i,\quad y^1_{i_0,j_0}=1,\ \forall j\in S^1,\ y^1_{i_0,j}=y^1_{j_0,j}=1,\ \forall (i,j)\in E\setminus(S^<\sqcup S:S^1),\ y^1_{i,j}=y_{i,j}$$

$$x^0_{i_0}=0,\ x^0_{j_0}=0,\ \forall i\in V\setminus S,\ x^0_i=x_i,\quad y^1_{i_0,j_0}=1,\ \forall j\in S^1,\ y^0_{i_0,j}=y^0_{j_0,j}=0,\ \forall (i,j)\in E\setminus(S^<\sqcup S:S^1),\ y^0_{i,j}=y_{i,j}$$

Using the tables on the right that sum up the values of z, z^1, and z^0, one can easily check that $z=\frac{z^1+z^0}{2}$.

Moreover, using Lemma 3.1, one can also check that $z^1\in S^n_{x,y}\subseteq QP^n_{\mathrm{LP}}$ and $z^0\in S^n_{x,y}\subseteq QP^n_{\mathrm{LP}}$.

Then z is the middle of two other points in QP^n_{LP}. A contradiction.

i	x_i	x^1_i	x^0_i
$\{i_0\}$	1/2	1	0
$\{j_0\}$	1/2	1	0
S^0	0	0	0
S^1	1	1	1

e	y_e	y^1_e	y^0_e
$\{i_0,j_0\}$	1/2	1	0
$\{i_0\}:S^0$	0	0	0
$\{i_0\}:S^1$	1/2	1	0
$\{j_0\}:S^0$	0	0	0
$\{j_0\}:S^1$	1/2	1	0
$(S^0)^<$	0	0	0
$S^0:S^1$	0	0	0
$(S^1)^<$	1	1	1

Finally, $|S|\notin\{1,2\}$, then either $S=\emptyset$ or $|S|\geqslant 3$. □

Lemma 3.7

If $z=(x,y)\in\mathrm{extr}\,(QP^n_{LP})$ *such that, for* $S=\left\{i\in V\,\middle|\,x_i=\frac{1}{2}\right\}$, $|S|\geqslant 3$,
then *there exist* $C\in\mathcal{C}y(K_n)$ *and* $F\subseteq C$ *with* $|F|$ *odd such that* z *does not satisfies* $(3.12)_{C,F}$.

Proof: Since $|S|\geqslant 3$ and K_n is complete, there exists $C\in\mathcal{C}y(K_n)$ that covers S, *i.e.* $\mathrm{supp}(C)=S$.
Note that for any $(i,j)\in S^<$, $y_{i,j}\in[0,\frac{1}{2}]$ from inequalities (3.1–3.2), thus $y_{i,j}\in\{0,\frac{1}{2}\}$ since z is extreme. We define F as the subset of 0-valued edges of C, *i.e.* $F=\{e\in C\,|\,y_e=0\}$. Then for any $e\in C\setminus F, y_e=\frac{1}{2}$. Let us partition the nodes of C, *i.e.* S, depending on how many incident C edges belong to $C\setminus F$, as done for inequalities (3.12), that is as follows.

$$S^0=\{i\in S\,|\,\exists(e,f)\in F\times F\text{ s.t. }e\neq f,\ i\in e\cap f\}$$
$$S^1=\{i\in S\,|\,\exists(e,f)\in (C\setminus F)\times F\text{ s.t. }e\neq f,\ i\in e\cap f\}$$
$$S^2=\{i\in S\,|\,\exists(e,f)\in (C\setminus F)\times (C\setminus F)\text{ s.t. }e\neq f,\ i\in e\cap f\}$$

By counting, for each node on C, how many incident edges on C belong to F, each edge in F is counted twice. Therefore, we have the following equation.

$$2|F| = |S^0|\times 2 + |S^1|\times 1 + |S^2|\times 0 = |S^0| + (\underbrace{|S^0|+|S^1|+|S^2|}_{=|S|}) - |S^2|$$

Since C is a cycle and S is its support, we have $|C|=|S|$. Moreover, since $F\subseteq C$, $|C|=|C\setminus F|+|F|$. We deduce that $2|F| = |S^0|-|S^2| + |C\setminus F|+|F|$, that is $|S^0|-|S^2| + |C\setminus F|=|F|$.

Therefore, using the definitions of S^0, S^2, and F we have the following equation.

$$\underbrace{x(S^0) - x(S^2)}_{=\frac{1}{2}(|S^0|-|S^2|)} + \underbrace{y(C\setminus F)}_{=\frac{1}{2}|C\setminus F|} - \underbrace{y(F)}_{=0} \;=\; \frac{1}{2}\Big(|S^0|-|S^2| + |C\setminus F|\Big) \;=\; \frac{|F|}{2}$$

• If $|F|$ is odd, $\left\lfloor \frac{|F|}{2} \right\rfloor = \frac{|F|}{2} - 1$, then $\frac{|F|}{2} > \left\lfloor \frac{|F|}{2} \right\rfloor$.
The previous equation shows that z does not satisfy the inequality (3.12) associated with C and F.

• If $|F|$ is even, $\left\lfloor \frac{|F|}{2} \right\rfloor = \frac{|F|}{2}$, then we cannot similarly conclude.
We will exhibit another cycle on $\frac{1}{2}$-valued nodes that contains an even number of 0-valued edges.
Note that, since $|F|$ is even, the numbers of F edges in the two paths along C between two nodes of S have the same parity, *i.e.* either both are even or both are odd, since their sum is $|F|$. This remark allows to enunciate the following claim without ambiguity.

Claim

> There exists $\{T, U\}$ a bi-partition of S such that, for any $(s, s')\in S\times S$, s and s' are in the same part if and only if the number of F edges on a path along C between s and s' is even

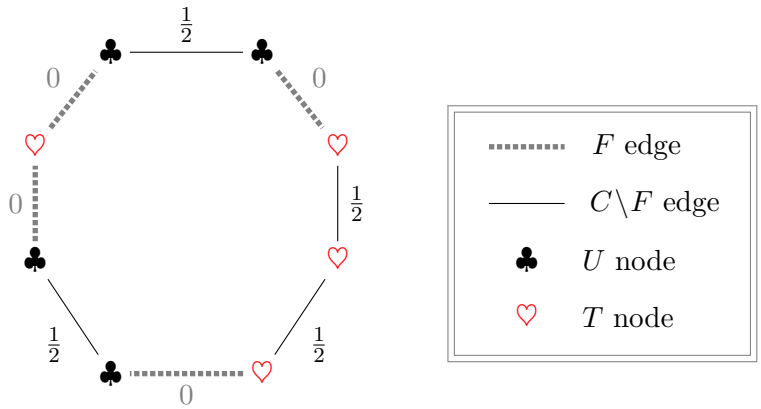
Proof: Let us denote by $(s_k)_{k\in[1..K]}$ the nodes in S such that

$$C = \{\{s_k, s_{k+1}\} \,|\, k \in [1..K-1]\} \cup \{\{s_K, s_1\}\}$$

To formally build $\{T, U\}$, we recursively assign s_k to T or U according to the following rules.

$$s_1 \in U \quad \text{and} \quad \forall k \in [1..K-1], \begin{cases} s_{k+1}\in U \text{ if } s_k \in U \text{ and } \{s_k, s_{k+1}\}\notin F \\ s_{k+1}\in U \text{ if } s_k \in T \text{ and } \{s_k, s_{k+1}\}\in F \\ s_{k+1}\in T \text{ otherwise} \end{cases}$$

This procedure is equivalent to color S nodes using two colors, starting from an arbitrary point on C, and going along C by changing color at each F edge. The following figure is provided to illustrate this construction for $|F|=4$.



One can check by induction on path length, that any path along C between two nodes s and s' contains an even number of edges in F if and only if $(s, s')\in(U\times T)\cup(T\times U)$. ∎

Since S is partitioned into $T \sqcup U$, the edges in $S^<$ are partitioned into $T^< \sqcup U^< \sqcup T{:}U$.
Let us recall that for any $e \in S^<$, $y_e \in \{0, \frac{1}{2}\}$.

Let us assume that $\begin{cases} \forall e \in T{:}U,\ y_e = 0 \\ \forall e \in T^< \sqcup U^<,\ y_e = \frac{1}{2} \end{cases}$ then we define $z^1 = (x^1, y^1)$ and $z^2 = (x^2, y^2)$ as follows.

$\forall u \in U, x^1_u = 0,\ \forall t \in T, x^1_t = 1,\ \forall i \in V \backslash S, x^1_i = x_i,\quad \forall e \in U^<, y^1_e = 0,\ \forall e \in T^<, y^1_e = 1,\ \forall e \in E \backslash (U^< \sqcup T^<), y^1_e = y_e$

$\forall u \in U, x^2_u = 1,\ \forall t \in T, x^2_t = 0,\ \forall i \in V \backslash S, x^2_i = x_i,\quad \forall e \in U^<, y^2_e = 1,\ \forall e \in T^<, y^2_e = 0,\ \forall e \in E \backslash (U^< \sqcup T^<), y^2_e = y_e$

One can easily check that $z = \frac{z^1 + z^2}{2}$.
Moreover, using Lemma 3.1, one can also check that $z^1 \in S^n_{x,y} \subseteq QP^n_{\text{LP}}$ and $z^2 \in S^n_{x,y} \subseteq QP^n_{\text{LP}}$.

Then z is the middle of two other points in QP^n_{LP}. A contradiction.

We deduce that there necessarily exits either $e \in T^<$ such that $y_e = 0$ or $e \in U^<$ such that $y_e = 0$, or $e \in U{:}T$ such that $y_e = \frac{1}{2}$.

– Let us assume that $e \in U{:}T$ such that $y_e = \frac{1}{2}$. By definition of $\{T, U\}$, we have $e \notin C$.
Indeed, if $e \in C$, we have $e \in C \setminus M$ since $y_e \neq 0$. Then $\{e\}$ is a path of length 1 between a node of U and a node of T that contains 0 M-edge. A contradiction since 0 is even.
Let us denote by μ one of the two paths along C between both e endpoints.
We define $C' = \mu \cup \{e\}$ and $M' = \{f \in C' \mid y_f = 0\}$. Note that $\text{supp}(C') \subseteq S$ and $M' \subseteq C'$.
Since $y_e \neq 0$, $M' = M \cap \mu$. Therefore, by definition of $\{T, U\}$, $|M'|$ is odd.
Hence, we can use for (C', M') the same arguments as for (C, M) previously.
We deduce that z does not satisfy the inequality (3.12) associated with C' and M'.

– Let us assume that $e \in U^<$ such that $y_e = 0$. By definition of $\{T, U\}$, we have $e \notin C$.
Indeed, if $e \in C$, we have $e \in M$ since $y_e = 0$. Then $\{e\}$ is a path of length 1 between two nodes of U that contains 1 M-edge. A contradiction since 1 is odd.
Let us denote by μ one of the two paths along C between both e endpoints.
We define $C' = \mu \cup \{e\}$ and $M' = \{f \in C' \mid y_f = 0\}$. Note that $\text{supp}(C') \subseteq S$ and $M' \subseteq C'$.
Since $y_e = 0$, $M' = M \cap \mu \sqcup \{e\}$. We deduce that $|M'|$ is odd, since $|M \cap \mu|$ is even, by definition of $\{T, U\}$.
Hence, we can use for (C', M') the same arguments as for (C, M) previously.
We deduce that z does not satisfy the inequality (3.12) associated with C' and M'.

– If $e \in T^<$ is such that $y_e = 0$, we conclude similarly. □

By combining Lemmas 3.6 and 3.7, we deduce that all the fractional extreme points of QP^n_{LP} are cut off by the odd sub-cycle inequalities. The following property states that these inequalities can be obtained by linear combination from triangle inequalities, therefore, the fractional extreme points of QP^n_{LP} are even cut off by triangles inequalities.

Property 3.8 (*Padberg [31]*)

Let $C \subseteq \mathcal{C}y(K_n)$ and $M \subseteq C$ such that $|M|$ is odd.
If *$|C| = 3$,* ***then*** *$(3.12)_{C,M}$ is one of the triangle inequalities (3.5–3.8) for the three nodes of C.*
If *$|C| > 3$,* ***then*** *$(3.12)_{C,M}$ is a linear combination of the triangle inequalities (3.5–3.8).*

Corollary 3.9

All the fractional extreme points of QP^n_{LP} are cut off by triangle inequalities (3.5–3.8).

3.4 Facets transposition

In this section, we propose a property to transpose valid or facet defining inequalities from a polyhedron to another. The proof is based on elementary properties of convex hull and polyhedron that are recalled in appendix, in Section CA.4. Note that this property only gives a formal synthesis of a common knowledge. We present it since we have not found an explicit and ready-to-use writing of this property. As it is enunciated below, Property 3.10 defines a framework that covers different results, such as symmetry or lifting properties. The second part of this section proposes several application of Property 3.10.

- *What means transposing inequalities*

Let assume that we have an affine map ψ that maps *points* of a space $\mathcal{E}^1$ to a space $\mathcal{E}^2$.
We are interested in a way to transform any inequality (I') of $\mathcal{E}^2$ into an inequality (I) of $\mathcal{E}^1$ such that they are "equivalent for ψ", that is $\forall X \in \mathcal{E}^1, X$ satisfies $(I) \Leftrightarrow \psi(X)$ satisfies (I').
Let us denote this "equivalence" by $(I) \overset{\psi}{\Leftrightarrow} (I')$.

To transform $\mathcal{E}^2$ inequalities into $\mathcal{E}^1$ inequalities, we use two functions: φ, which maps *vectors* of $\mathcal{E}^2$ to those of $\mathcal{E}^1$, and f which map vectors of $\mathcal{E}^2$ to *values*. For any $(a, \alpha) \in \mathcal{E}^2 \times \mathbb{R}$, the inequality $(I') : a \cdot Y \leqslant \alpha$ of $\mathcal{E}^2$ is transformed into the inequality $(I) : \varphi(a) \cdot X \leqslant \alpha - f(a)$ of $\mathcal{E}^1$.
Let us denote this transformation by $(I) \xrightarrow{\varphi, f} (I')$, and called it **transposition** in the sequel.

Property 3.10 states that, under some assumption $(\bigstar)$, φ and f transpose valid (resp. facet defining) inequalities for $\psi(P)$ into valid (resp. facet defining) inequalities for P any polyhedron of $\mathcal{E}^1$. This assumption means that transposing inequalities using (φ, f) provides equivalent inequalities for ψ.
Let us synthesized it as follows. $(\bigstar)$: **If** $(I) \xrightarrow{\varphi, f} (I')$, **then** $(I) \overset{\psi}{\Leftrightarrow} (I')$.

3.4.1 A practical property to transpose valid or facet inequalities

Property 3.10

Let $\mathcal{E}^1$ and $\mathcal{E}^2$ two Euclidian spaces. Let ψ an affine map from $\mathcal{E}^1$ to $\mathcal{E}^2$.
Let $S^1 \subseteq \mathcal{E}^1$, $P^1 = \text{conv}(S^1)$, $S^2 = \psi(S^1)$ and $P^2 = \text{conv}(S^2)$.

***If** there exist $\varphi \in \mathcal{F}(\mathcal{E}^2, \mathcal{E}^1)$ and $f \in \mathcal{F}(\mathcal{E}^2, \mathbb{R})$ such that $\forall X \in \mathcal{E}^1, \forall a \in \mathcal{E}^2$, $a \cdot \psi(X) = \varphi(a) \cdot X + f(a)$ $(\bigstar)$,*
***then** for any $a \in \mathcal{E}^2$ and any $\alpha \in \mathbb{R}$ we have:*
(i) $a \cdot Y \leqslant \alpha$ is a valid inequality for $P^2 \Leftrightarrow \varphi(a) \cdot X \leqslant \alpha - f(a)$ is a valid inequality for P^1.
(ii) $a \cdot Y \leqslant \alpha$ defines a face of $P^2 \Leftrightarrow \varphi(a) \cdot X \leqslant \alpha - f(a)$ defines a face of P^1.

***If** in addition to $(\bigstar)$, we assume that $\dim(P^1) = \dim(P^2)$,*
***then** for any $a \in \mathcal{E}^2$ and any $\alpha \in \mathbb{R}$ we have:*
(iii) $a \cdot Y \leqslant \alpha$ defines a facet of $P^2 \Rightarrow \varphi(a) \cdot X \leqslant \alpha - f(a)$ defines a facet of P^1.

***If** in addition to $(\bigstar)$, we assume that ψ is bijective,*
***then** $\dim(P^1) = \dim(P^2)$ and for any $a \in \mathcal{E}^2$ and any $\alpha \in \mathbb{R}$ we have:*
(iv) $a \cdot Y \leqslant \alpha$ defines a facet of $P^2 \Leftrightarrow \varphi(a) \cdot X \leqslant \alpha - f(a)$ defines a facet of P^1.

Proof: Let $a \in \mathcal{E}^2$ and $\alpha \in \mathbb{R}$.
Let us denote by H^1 the hyperplane of $\mathcal{E}^1$ defined by the inequality $\varphi(a) \cdot X \leqslant \alpha - f(a)$ and by $H^{1>} \subseteq E$ the set of points that do not satisfy this inequality (*nb*: $H^{1>}$ is not an half-space, but the complement of an half-space).

$$H^1 = \{X \in \mathcal{E}^1 \mid \varphi(a) \cdot X = \alpha - f(a)\} \quad \text{and} \quad H^{1>} = \{X \in \mathcal{E}^1 \mid \varphi(a) \cdot X > \alpha - f(a)\}$$

We introduce similarly H^2 and $H^{2>}$ for the inequality $a\cdot Y \leqslant \alpha$ in $\mathcal{E}^2$.

$$H^2 = \{Y \in \mathcal{E}^2 \mid a\cdot Y = \alpha\} \quad \text{and} \quad H^{2>} = \{Y \in \mathcal{E}^2 \mid a\cdot Y > \alpha\}$$

Since ψ is linear, we have $\psi(\operatorname{conv}(S^1)) = \operatorname{conv}(\psi(S^1)) = \operatorname{conv}(S^2)$, then $\psi(P^1) = P^2$.

Moreover, using the assumption (★), one can check that $\psi(H^1) = H^2$ and $\psi(H^{1>}) = H^{2>}$.

Let us show that $\psi(S^1 \cap H^1) = S^2 \cap H^2$.

By definition of the image of a subset, we have $\psi(S^1 \cap H^1) \subseteq \psi(S^1) \cap \psi(H^1)$.

Then, according to the previous remark, we have $\psi(S^1 \cap H^1) \subseteq S^2 \cap H^2$.

Let us show the reverse inclusion. Let $Y \in S^2 \cap H^2$.

Since $S^2 = \psi(S^1)$, there exists $X \in S^1$ such that $Y = \psi(X)$.

By definition of H^2, we have $a\cdot Y = \alpha$, that is $a\cdot\psi(X) = \alpha$.

By assumption (★), we deduce that $\varphi(a)\cdot X + f(a) = \alpha$.

Hence $X \in H^1$, and then $X \in H^1 \cap S^1$. Thus, $Y \in \psi(H^1 \cap S^1)$. This shows that $S^2 \cap H^2 \subseteq \psi(S^1 \cap H^1)$.

Finally we have $S^2 \cap H^2 = \psi(S^1 \cap H^1)$.

Similarly, one can show that $\psi(P^1 \cap H^1) = P^2 \cap H^2$ and $\psi(P^1 \cap H^{1>}) = P^2 \cap H^{2>}$.

• By definition, $a\cdot Y \leqslant \alpha$ is valid for P^2 if and only if $P^2 \cap H^{2>} = \emptyset$, that is if $\psi(P^1 \cap H^{1>}) = \emptyset$.
Similarly, $\varphi(a)\cdot X \leqslant \alpha - f(a)$ is valid for P^1 if and only if $P^1 \cap H^{1>} = \emptyset$, which is equivalent to $\psi(P^1 \cap H^{1>}) = \emptyset$ by definition of the image of a subset. We deduce that *(i)* is true.

• By definition, $a\cdot Y \leqslant \alpha$ defines a face of P^2 if and only if $H^{2>} \cap P^2 = \emptyset$ and $H^2 \cap P^2 \neq \emptyset$.
Similarly, $\varphi(a)\cdot X \leqslant \alpha - f(a)$ defines a face of P^1 if and only if $H^{1>} \cap P^1 = \emptyset$ and $H^1 \cap P^1 \neq \emptyset$.
Therefore, since *(i)* is true, to prove *(ii)* it suffices to show that $H^2 \cap P^2 \neq \emptyset \Leftrightarrow H^1 \cap P^1 \neq \emptyset$.
Since we have $\psi(P^1 \cap H^1) = P^2 \cap H^2$, this is true by definition of the image of a subset.
Thus, *(ii)* is true.

• Let us denote by d the dimension of both P^1 and P^2.
By definition, $a\cdot Y \leqslant \alpha$ defines a facet of P^2 if and only if $H^{2>} \cap P^2 = \emptyset$ and $\dim(H^2 \cap P^2) = d-1$.
Similarly, $\varphi(a)\cdot X \leqslant \alpha - f(a)$ defines a facet of P^1 if and only if $H^{1>} \cap P^1 = \emptyset$ and $\dim(H^1 \cap P^1) = d-1$.
Therefore, since *(i)* is true, to prove *(iii)* it suffices to show that $\dim(H^2 \cap P^2) = d-1 \Rightarrow \dim(H^1 \cap P^1) = d-1$.
According to Corollary CA.57*(ii)*, it is equivalent to show that $\dim(H^2 \cap S^2) = d-1 \Rightarrow \dim(H^1 \cap S^1) = d-1$.
Let assume that $\dim(S^2 \cap H^2) = d-1$.
Since $S^2 \cap H^2 = \psi(S^1 \cap H^1)$, by Corollary CA.53 we have $d-1 = \dim(S^2 \cap H^2) \leqslant \dim(S^1 \cap H^1)$.
Since $\dim(S^1 \cap H^1) \leqslant \dim(S^1) = \dim(P^1) = d$, we then have $\dim(S^1 \cap H^1) \in \{d-1, d\}$.
Let assume that $\dim(S^1 \cap H^1) = d = \dim(P^1) = \dim(S^1)$.
The inclusion $\operatorname{aff}(S^1 \cap H^1) \subseteq \operatorname{aff}(S^1)$ is then an equality, *i.e.* $\operatorname{aff}(S^1 \cap H^1) = \operatorname{aff}(S^1)$.
Then we have $S^1 \subseteq \operatorname{aff}(S^1) = \operatorname{aff}(S^1 \cap H^1) \subseteq \operatorname{aff}(H^1) = H^1$, and thus $S^2 = \psi(S^1) \subseteq \psi(H^1) = H^2$.
The latter inclusion implies that $S^2 \cap H^2 = S^2$, and thus $\dim(S^2 \cap H^2) = \dim(S^2) = \dim(P^2) = d$.
A contradiction. Then necessarily $\dim(S^1 \cap H^1) = d-1$.

• If ψ is bijective, then according to Corollary CA.53*(ii)* we have $\dim(\psi(S^1)) = \dim(S^1)$, and since $\psi(S^1) = S^2$, we deduce that $\dim(S^2) = \dim(S^1)$. It follows that $\dim(P^2) = \dim(P^1)$.
Similarly, $\dim(P^1 \cap H^1) = \dim(\psi(P^1 \cap H^1))$, then $\dim(P^1 \cap H^1) = \dim(P^2 \cap H^2)$ since $\psi(P^1 \cap H^1) = P^2 \cap H^2$.
Therefore, $\dim(P^1 \cap H^1) = \dim(P^1) - 1 \Leftrightarrow \dim(P^2 \cap H^2) = \dim(P^2) - 1$, and *(iv)* follows. □

In the sequel we provide different examples where inequalities are transposed from a polyhedron P to a polyhedron P', and then where Property 3.10 allows to derive facet defining inequalities for P from those for P'. For each example, we proceed in three steps:

→ *identify an affine map ψ which map of P to P',*

→ *identify the function φ and f such that (★) is satisfied,*

→ *transpose the inequalities using φ and f.*

3.4.2 Application to QP^n and CUT^{n+1}

In [12], De Simone establishes the link between the Boolean quadric polytope for an arbitrary graph G, and the cut polytope of the graph obtained by adding to G a node together with an edge from this node to each other node. Let us present this link for the particular case of complete graph using the framework of Property 3.10.

• *Notations*

We denote by $K_{n+1}=(\widetilde{V},\widetilde{E})$ the complete undirected graph on $[1..n+1]$, which corresponds to the following sets.

$$\widetilde{V}=[1..n+1]=V\sqcup\{n+1\} \quad \text{and} \quad \widetilde{E}=\widetilde{V}^{<}=E\sqcup\Big(V:\{n+1\}\Big)$$

Moreover, for any $S\subseteq\widetilde{V}$, we define $\widetilde{\chi}^S\in\mathbb{R}^{\widetilde{E}}$ as $\widetilde{\boldsymbol{\chi}}^{\boldsymbol{S}}=\mathbb{I}_{S:\widetilde{V}\setminus S}$. This way we have $CUT^{n+1}=\operatorname{conv}\Big(\Big\{\widetilde{\chi}^S\,\Big|\,S\subseteq\widetilde{V}\Big\}\Big)$.

• *Identifying the affine map ψ*

Let us define the **covariance map** $\widetilde{\psi}$ and its inverse function $\widetilde{\psi}^{-1}$ as follows.

$$\widetilde{\boldsymbol{\psi}}=\begin{pmatrix}\mathbb{R}^V\times\mathbb{R}^E & \longrightarrow & \mathbb{R}^{\widetilde{E}}\\ (x,y) & \longmapsto & X\end{pmatrix} \text{ where } \forall(i,j)\in\widetilde{E},\ X_{i,j}=\begin{cases}x_i+x_j-2\,y_{i,j} & \text{if } (i,j)\in E\\ x_i & \text{if } j=n+1\end{cases}$$

$$\widetilde{\boldsymbol{\psi}}^{\boldsymbol{-1}}=\begin{pmatrix}\mathbb{R}^{\widetilde{E}} & \longrightarrow & \mathbb{R}^V\times\mathbb{R}^E\\ X & \longmapsto & (x,y)\end{pmatrix} \text{ where } \begin{cases}\forall i\in V,\ x_i=X_{i,n+1}\\ \forall(i,j)\in E,\ y_{i,j}=\dfrac{-X_{i,j}+X_{i,n+1}+X_{j,n+1}}{2}\end{cases}$$

One can check that $\widetilde{\psi}$ is a bijective linear map and that $\forall S\subseteq V,\ \widetilde{\psi}(z^S)=\widetilde{\chi}^S$.

We deduce that $\widetilde{\psi}(S^n_{x,y})=S^{n+1}_X$ and $\boxed{\widetilde{\psi}(QP^n)=CUT^{n+1}}$.

• *Identifying functions "φ and f"*

By rewriting the expression $a\cdot\widetilde{\psi}(x,y)$ as a weighted sum of variables x and y, we identify the coefficient of a vector b such that $a\cdot\widetilde{\psi}(z)=b\cdot(x,y)$. Therefore, we introduce the following function $\widetilde{\varphi}$.

$$\widetilde{\boldsymbol{\varphi}}=\begin{pmatrix}\mathbb{R}^{\widetilde{E}} & \longrightarrow & \mathbb{R}^V\times\mathbb{R}^E\\ a & \longmapsto & b\end{pmatrix} \text{ where } \begin{cases}\forall i\in V,\ b_i=a\Big(\{i\}:\widetilde{V}\setminus\{i\}\Big)\\ \forall e\in E,\ b_e=-2\,a_e\end{cases}$$

By construction of $\widetilde{\varphi}$, we have $\forall a\in\mathbb{R}^{\widetilde{E}},\forall z\in\mathbb{R}^V\times\mathbb{R}^E,\ a\cdot\widetilde{\psi}(z)=\widetilde{\varphi}(a)\cdot z$.
To stick to the Property 3.10's framework, we could set additionally $\widetilde{f}=0$.

• *Transposing inequalities using "φ and f"*

Transposition 3.11 (*Triangle and trivial inequalities*)

Let $(i,j,k)\in V^{<<}$.

$$\textbf{\textit{If }}k\neq n+1,\quad \textbf{\textit{then}}\quad \left.\begin{matrix}(3.13)_{i,j,k}\\ (3.14)_{i,j,k}\\ (3.15)_{i,j,k}\\ (3.16)_{i,j,k}\end{matrix}\right\}\xrightarrow{\widetilde{\varphi},0}\left\{\begin{matrix}(3.5)_{i,j,k}\\ (3.6)_{i,j,k}\\ (3.7)_{i,j,k}\\ (3.8)_{i,j,k}\end{matrix}\right. \quad \textbf{\textit{otherwise}}\quad \left.\begin{matrix}(3.13)_{i,j,k}\\ (3.14)_{i,j,k}\\ (3.15)_{i,j,k}\end{matrix}\right\}\xrightarrow{\widetilde{\varphi},0}\left\{\begin{matrix}(3.2)_{i,j}\\ (3.3)_{i,j}\\ (3.4)_{i,j}\end{matrix}\right.$$

$$(3.16)_{i,j,k}\xrightarrow{\widetilde{\varphi},0}(3.1)_{i,j}$$

Triangles inequalities for CUT^{n+1} are equivalent to both trivial and triangle inequalities for QP^n.

Transposition 3.12 (*Odd sub-cycle inequalities*)

Let $C \in \mathcal{C}y(K_{n+1})$. Let $S = \text{supp}(C)$. Let F be an odd sub-cycle of C, i.e. $F \subseteq C$ s.t. $|F|$ is odd.

If $n+1 \notin S$, ***then*** $(3.17)_{C,F} \overset{\widetilde{\psi}}{\Leftrightarrow} (3.12)_{C,F}$.

Otherwise, *let us denote by $e = \{k, n+1\}$ and $e' = \{k', n+1\}$ the two edges in C incident to $n+1$. Let us introduce the edge $f = \{k, k'\}$ and the cycle $C' = C \cup \{f\} \setminus \{e, e'\}$. We then have:*

- *if $e \in F$ and $e' \in F$, $(3.17)_{C,F} \overset{\widetilde{\psi}}{\Leftrightarrow} (3.12)_{C',F'} + (3.1)_{k,k'}$ for $F' = F \setminus \{e, e'\}$*
- *if $e \notin F$ and $e' \in F$, $(3.17)_{C,F} \overset{\widetilde{\psi}}{\Leftrightarrow} (3.12)_{C',F'} + (3.2)_{k,k'}$ for $F' = F \cup \{f\} \setminus \{e'\}$*
- *if $e \in F$ and $e' \notin F$, $(3.17)_{C,F} \overset{\widetilde{\psi}}{\Leftrightarrow} (3.12)_{C',F'} + (3.3)_{k,k'}$ for $F' = F \cup \{f\} \setminus \{e\}$*
- *if $e \notin F$ and $e' \notin F$, $(3.17)_{C,F} \overset{\widetilde{\psi}}{\Leftrightarrow} (3.12)_{C',F'} + (3.4)_{k,k'}$ for $F' = F$*

Odd sub-cycle inequalities for CUT^{n+1} are equivalent to a sum of an odd sub-cycle inequalities and a trivial inequality for QP^n.

Transposition 3.13 (*Clique inequalities*)

Let $S \subseteq V$ such that $|S|$ is odd.

If $n+1 \notin S$, ***then*** $(3.18)_S \overset{\widetilde{\psi}}{\Leftrightarrow} (3.9)_{S,\theta}$ *for* $\theta = \left\lfloor \frac{|S|}{2} \right\rfloor$, ***otherwise*** $(3.18)_S \overset{\widetilde{\psi}}{\Leftrightarrow} (3.9)_{S',\theta'}$ *for* $\begin{cases} S' = S \setminus \{n+1\} \\ \theta' = \left\lfloor \frac{|S'|}{2} \right\rfloor \end{cases}$

Clique inequalities over an odd subset for CUT^{n+1} are transposed into clique inequalities for QP^n.

Note that the clique inequalities for QP^n as defined by Padberg in [31], *i.e.* (3.9) are a generalization of the clique inequalities for CUT^{n+1}, *i.e.* (3.18). For example, inequalities (3.9) for a parameter θ that is not $\left\lfloor \frac{|S|}{2} \right\rfloor$ are not the transposition of any inequality (3.18).

3.4.3 Application to QP^n

In this section, we do not transpose inequalities from one polyhedron to another, but from a polyhedron to itself. Indeed, for a given $M \subseteq V$, Padberg introduces in [31] an affine transformation of $\mathbb{R}^V \times \mathbb{R}^E$ which maps QP^n to itself. Such a transformation can be seen as a symmetry of QP^n, or even as a symmetry on the set of ordered bi-partitions. Let us consider a partition (S,T). By changing the part to which each node of M belongs, we obtain the partition $(S \triangle M, T \triangle M)$, and by applying again the same operation, we obtain back (S,T). Therefore, this operation is a kind of symmetry over the ordered bi-partitions. The equivalent of this operation on vectors (x,y) that represent partitions is the map ψ^M introduced below.

- *Identifying the affine map ψ*

$$\boldsymbol{\psi^M} = \begin{pmatrix} \mathbb{R}^V \times \mathbb{R}^E & \longrightarrow & \mathbb{R}^V \times \mathbb{R}^E \\ (x,y) & \longmapsto & (x',y') \end{pmatrix} \text{ where } \begin{cases} \forall i \in V,\ x'_i = \begin{cases} 1 - x_i & \text{if } i \in M \\ x_i & \text{otherwise} \end{cases} \\ \\ \forall (i,j) \in E,\ y_{i,j} = \begin{cases} -y_{i,j} + x_j & \text{if } i \in M \text{ and } j \notin M \\ 1 - (y_{i,j} + x_i + x_j) & \text{if } i \in M \text{ and } j \in M \\ y_{i,j} & \text{otherwise} \end{cases} \end{cases}$$

One can check that ψ^M is an affine map and that $\forall S \in V,\ \psi(z^S) = z^{S \triangle M}$, where $S \triangle M$ denotes the symmetric difference of sets S and M. The latter point shows that the image of ψ^M is fully dimensional, since the points z^S for all $S \subseteq V$ generate $\mathbb{R}^V \times \mathbb{R}^E$, and any $S \subseteq V$ can be written as $(S \triangle M) \triangle M$. Since ψ^M is an affine map of $\mathbb{R}^V \times \mathbb{R}^E$ whose image is $\mathbb{R}^V \times \mathbb{R}^E$, it is even an affine transformation. Moreover, this point shows that $\boxed{\psi^M(QP^n) = QP^n}$.

- *Identifying functions "φ and f"*

As previously, by rewriting the expression $a \cdot \psi^M(x,y)$ as a weighted sum of variables x and y, we identify the coefficient of a vector b such that $a \cdot \psi^M(x,y) = b \cdot (x,y) + c$ where c is a constant, in the sense that it does not depends on (x,y). This way, we introduce the following functions φ^M and f^M.

$$\boldsymbol{\varphi^M} = \begin{pmatrix} \mathbb{R}^V \times \mathbb{R}^E & \longrightarrow & \mathbb{R}^V \times \mathbb{R}^E \\ a & \longmapsto & b \end{pmatrix} \text{ where } \begin{cases} \forall i \in V,\ b_i = \begin{cases} -a_i - a\big(i : M \setminus \{i\}\big) & \text{if } i \in M \\ a_i + a(i : M) & \text{otherwise} \end{cases} \\ \\ \forall e \in E,\ b_e = \begin{cases} -a_e \text{ if } e \in M : V \setminus M \\ a_e \text{ otherwise} \end{cases} \end{cases}$$

$$\boldsymbol{f^M} = \begin{pmatrix} \mathbb{R}^V \times \mathbb{R}^E & \longrightarrow & \mathbb{R} \\ a & \longmapsto & a(M) + a(M^<) \end{pmatrix}$$

By construction of φ^M and f^M, we have $\forall a \in \mathbb{R}^V \times \mathbb{R}^E, \forall z \in \mathbb{R}^V \times \mathbb{R}^E,\ a \cdot \psi^M(z) = \varphi^M(a) \cdot z + f^M(a)$.

- *Transposing inequalities using φ and f*

Transposition 3.14 (*Clique inequalities*)

Let $S \subseteq V$ such that $|S| \geqslant 2$, and $\alpha \in [1..|S|-2]$. We have $(3.9)_{S,\alpha} \xrightarrow{\varphi^V, f^V} (3.9)_{S,\beta}$ where $\beta = |S| - 1 - \alpha$. The clique inequalities are stable for the transposition using φ^V and f^V.

Transposition 3.15 (*Cut inequalities*)

Let $(S,T) \subseteq \mathcal{P}(V)^2$ such that $S \cap T \neq \emptyset, |S| \geqslant 1$, and $|T| \geqslant 2$. We have $(3.10)_{S,T} \xrightarrow{\varphi^V, f^V} (3.11)_{S,T}$. The generalized cut inequalities for QP^n are equivalent to the cut inequalities for QP^n.

Actually, generalized cut inequalities were obtained by transposing cut inequalities using (φ^V, f^V) (*Cf.* [31]).

• *A note on about the Padberg's symmetry theorem in [31]*

Padberg [31] introduced functions ψ^M, φ^M and f^M (under slightly different notations) and use them to prove the following "Symmetry theorem" (Theorem 6 in [31]).

> *For any $R \subseteq V$ and $S \subseteq V$, and any facet $a \cdot z \leqslant \alpha$ of QP^n such that $a \cdot z^R = \alpha$, there exists a unique facet $\tilde{a} \cdot z \leqslant \tilde{\alpha}$ of QP^n such that $\tilde{a} \cdot z^S = \tilde{\alpha}$.*

As enunciated, this theorem is false. Indeed, since the extreme points of QP^n are exactly the points z^S for $S \subseteq V$, this theorem states that for any pair of extreme points z^S and z^R, and for any facet of QP^n containing z^R, there exists only one facet of QP^n containing z^S. That is not true.
Moreover, the existence or the unicity of a facet containing z^S does not depend on z^R or on a facet containing z^R. The point was surely to say that there exists only one facet obtained by "symmetry" from $a \cdot z \leqslant \alpha$ that contains z^R. However, the unicity does not stand, as shown by the counter-example given below. Therefore, let us provide a revised version of the symmetry theorem.

Property 3.16

> *Let $R \subseteq V$. Let $(a, \alpha) \in (\mathbb{R}^V \times \mathbb{R}^E) \times \mathbb{R}$ such that $a \cdot z \leqslant \alpha$ defines a facet of QP^n and $a \cdot z^R = \alpha$.*
> *(i) For any $M \subseteq V$, the inequality $\varphi^M(a) \cdot z \leqslant \alpha - f^M(a)$ defines a facet of QP^n.*
> *(ii) Moreover, for any $S \subseteq V$, if $M = R \triangle S$, then this facet contains z^S.*

Proof: Point (i) is exactly what ensures Property 3.10 *(iii)*. Besides, z^S belongs to the facet obtained using φ^M and f^M if and only if $\varphi^M(a) \cdot z^S = \alpha - f^M(a)$ and $\varphi^M(a) \cdot z^S + f^M(a) = \alpha \Leftrightarrow a\psi^M(z^S) = \alpha \Leftrightarrow a \cdot z^{S \triangle M} = a \cdot z^R$.
Therefore, if $M = S \triangle R$, then $S \triangle M = S \triangle (S \triangle R) = R$, and z^S belongs to the facet.
Note that conversely, $a \cdot z^{S \triangle M} = a \cdot z^R$ does not implies that $S \triangle M = R$. □

Counter-example 4: *Among the facets obtained by symmetry from an inequality satisfied by z^R, there can exist more than one that contains z^S.*

Let $(i, j, k) \in V^{<<}$. Let us set $S = \{i\}$, $R = \{j\}$, $M^0 = S \triangle R = \{i, j\}$ and $M = \{i, j, k\}$.
We have $S \triangle M = \{j, k\} = (S \triangle M^0) \sqcup \{k\}$.

Therefore, by denoting $\begin{cases} (x^{S\triangle M^0}, y^{S\triangle M^0}) = z^{S\triangle M^0} \\ (x^{S\triangle M}, y^{S\triangle M}) = z^{S\triangle M} \end{cases}$ we obtain $\begin{cases} x^{S\triangle M} = x^{S\triangle M^0} + \mathbb{I}_k \in \mathbb{R}^V \\ y^{S\triangle M} = y^{S\triangle M^0} + \mathbb{I}_{\{i,k\}} \in \mathbb{R}^E \end{cases}$

We set $a = (-\mathbb{I}_k, \mathbb{I}_{\{i,k\}})$, thus $y_{i,k} \leqslant x_k$ $(3.3)_{i,k}$, which is satisfied to equality by z^S, can be written $a.z \leqslant 0$.
Let us explicit below the components of $b^0 = \varphi^{M^0}$ and $b = \varphi^M$.

$b^0_i = -a_i - a_{i,j} = 0$	$b_i = -a_i - a_{i,j} - a_{i,k} = -1$
$b^0_j = -a_j - a_{i,j} = 0$	$b_j = -a_j - a_{i,j} - a_{j,k} = 0$
$b^0_k = +a_k + a_{i,k} + a_{j,k} = -1 + 1$	$b_k = -a_k - a_{i,k} - a_{j,k} = 0$
$\forall l \in V \setminus \{i, j, k\}$, $b^0_l = 0$	$\forall l \in V \setminus \{i, j, k\}$, $b_l = 0$
$\forall e \in E, e \neq \{i, k\}, a_e = -a_e = 0$ then $b^0_e = 0$	similarly $\forall e \in E, e \neq \{i, k\}, b_e = 0$
$b^0_{i,k} = -a_{i,k} = -1$ since $i \in M^0$ and $k \notin M^0$	$b_{i,k} = a_{i,k} = 1$ since $i \in M$ and $k \in M$
Finally, $b^0 = (0, -\mathbb{I}_{i,k})$.	Finally, $b = (-\mathbb{I}_i, -\mathbb{I}_{i,k})$.
Moreover, $f^{M^0}(a) = a(M^0) + a((M^0)^<) = a_i + a_{i,k} = 0$.	Moreover, $f^M(a) = a(M) + a(M^<) = a_i + a_{i,k} = 0$.
Then the inequality $b^0 \cdot z \leqslant f^{M^0}(a)$ is $-y_{i,k} \leqslant 0$,	Then the inequality $b \cdot z \leqslant f^M(a)$ is $-x_i + y_{i,k} \leqslant 0$,
i.e. $y_{i,k} \geqslant 0$ that is $(3.4)_{i,k}$	*i.e.* $y_{i,k} \leqslant x_i$ that is $(3.2)_{i,k}$

The point $z^R = z^{\{j\}} = (\mathbb{I}_j, 0)$ satisfies both $(3.4)_{i,k}$ and $(3.2)_{i,k}$ to equality. In other words, z^R belongs to two different facets of QP^n that can be obtained by different symmetries from the same inequality $(3.3)_{i,k}$.

3.4.4 Application to $P^n_{\delta,X}$

For any $M \subseteq V$, the symmetry operation on the ordered bi-partitions that consists in changing the part to which each node of M belongs, corresponds also to a symmetry over $P^n_{\delta,X}$. Therefore, we introduce below a function $\psi^M_{\delta,X}$ for $P^n_{\delta,X}$ which is analog to ψ^M for QP^n.

- *Identifying the affine map ψ*

$$\boldsymbol{\psi^M_{\delta,X}} = \begin{pmatrix} \mathbb{R}^V\times\mathbb{R}^E & \longrightarrow & \mathbb{R}^V\times\mathbb{R}^E \\ (\delta, X) & \longmapsto & (\delta',\, X') \end{pmatrix} \text{ where } \begin{cases} \forall i \in V,\ \delta'_i = \begin{cases} 1-\delta_i & \text{if } i \in M \\ \delta_i & \text{otherwise} \end{cases} \\ \\ \forall (i,j) \in E,\ X'_{i,j} = \begin{cases} 1-X_{i,j} & \text{if } \{i,j\} \in M : V \setminus M \\ X_{i,j} & \text{otherwise} \end{cases} \end{cases}$$

- *Identifying functions φ and f*

$$\boldsymbol{\varphi^M_{\delta,X}} = \begin{pmatrix} \mathbb{R}^V\times\mathbb{R}^E & \longrightarrow & \mathbb{R}^V\times\mathbb{R}^E \\ a=(a^\delta, a^X) & \longmapsto & (b^\delta, b^X) \end{pmatrix} \text{ where } \begin{cases} \forall i \in V,\ b^\delta_i = \begin{cases} -a_i & \text{if } i \in M \\ a_i & \text{otherwise} \end{cases} \\ \\ \forall (i,j) \in E,\ b^Y_{i,j} = \begin{cases} -a_{i,j} & \text{if } \{i,j\} \in M : V \setminus M \\ a_i & \text{otherwise} \end{cases} \end{cases}$$

$$\boldsymbol{f^M_{\delta,X}} = \begin{pmatrix} \mathbb{R}^V\times\mathbb{R}^E & \longrightarrow & \mathbb{R} \\ a & \longmapsto & a(M)+a(M:V\setminus M) \end{pmatrix}$$

One can check that we have $\forall a \in \mathbb{R}^V\times\mathbb{R}^E, \forall z \in \mathbb{R}^V\times\mathbb{R}^E,\ a \cdot \psi^M_{\delta,X}(z) = \varphi^M_{\delta,X}(a)\cdot z + f^M_{\delta,X}(a)$.

Note that, when $M = V$, the expressions of the previous functions can be simplified as follows.

$$\psi^V_{\delta,X} = (\delta, X) \mapsto (\mathbb{I}_V - \delta, X), \qquad \varphi^V_{\delta,X} = (a^\delta, a^X) \mapsto (-a^\delta, a^X), \qquad f^V_{\delta,X} = a \mapsto a(V)$$

- *Transposing inequalities using φ and f*

Even if $\psi^M_{\delta,X}(P^n_{\delta,X}) = P^n_{\delta,X}$, we do not explicitly transpose inequalities of $P^n_{\delta,X}$ in this section. However, transposition using $\varphi^M_{\delta,X}$ and $f^M_{\delta,X}$ is particularly useful when $M = V$, since for any $S \subseteq V$, $\psi^V_{\delta,X}$ maps the vector encoding the ordered bi-partition $(S, V\setminus S)$ to the vector encoding $(V\setminus S, S)$, *i.e.* $\psi^V_{\delta,X}(\mathbb{I}_S, \mathbb{I}_{S:V\setminus S}) = (\mathbb{I}_{V\setminus S}, \mathbb{I}_{S:V\setminus S})$.

In particular, $\psi^V_{\delta,X}$ maps the two vectors corresponding to the bi-partition $\{V, \emptyset\} = \{\emptyset, V\}$, and hence corresponding to the trivial cut, *i.e.* $\psi^V_{\delta,X}(0,0) = (\mathbb{I}_V, 0)$. Therefore, each inequality that cuts off $(0,0)$ can be transposed into an inequality cutting off $(\mathbb{I}_V, 0)$. The result is even more interesting. For any polyhedron invariant[1] under $\psi^V_{\delta,X}$, a facet defining inequality cutting off $(0,0)$ is transposed into a facet defining inequality cutting off $(\mathbb{I}_V, 0)$ using $\varphi^V_{\delta,X}$ and $f^V_{\delta,X}$.
We will use this result in the next chapter for $\tilde{\tilde{P}}^n_{\delta,X}$, the polytope of non-trivial cuts encoded by δ, X variables, which is stable by $\psi^V_{\delta,X}$.

[1]Note that in general, a set S is said **invariant** by a function ψ when $\psi(S) \subseteq S$.
In the particular case where ψ is a symmetry, *i.e.* $\psi \circ \psi = id$, this implies that $S = \psi\circ\psi(S) \subseteq \psi(S)$, and hence $\psi(S) = S$.

3.4.5 Application to QP^n and QP^N for $N > n$

In this section, we present the lifting theorem provided by Padberg in [31] as an application of Property 3.10. Let us consider $N \in \mathbb{N}$ such that $N > n$. We denote by $K_N = (\widetilde{V}, \widetilde{E})$ the complete graph on the node set $[1..N]$, which corresponds to $\widetilde{V} = [1..N]$ and $\widetilde{E} = \widetilde{V}^<$.

• *Identifying the affine map ψ*

The following mapping ψ^π is the projection of the space $\mathbb{R}^{\widetilde{V}} \times \mathbb{R}^{\widetilde{E}}$ on the space $\mathbb{R}^V \times \mathbb{R}^E$.

$$\boldsymbol{\psi^\pi} = \begin{pmatrix} \mathbb{R}^{\widetilde{V}} \times \mathbb{R}^{\widetilde{E}} & \longrightarrow & \mathbb{R}^V \times \mathbb{R}^E \\ (x, y) & \longmapsto & (x_{/V}, y_{/E}) \end{pmatrix}$$

As a projection, ψ^π is a linear map (not bijective).

Moreover, one can easily check that $\boxed{\psi^\pi(QP^N) = QP^n}$.

• *Identifying functions φ and f*

In order to have $\forall z \in \mathbb{R}^{\widetilde{V}} \times \mathbb{R}^{\widetilde{E}}, a \cdot \psi^\pi(z) = \varphi^\pi(a) \cdot z$, we define φ^π as the extension by 0 components.

$$\boldsymbol{\varphi^\pi} = \begin{pmatrix} \mathbb{R}^V \times \mathbb{R}^E & \longrightarrow & \mathbb{R}^{\widetilde{V}} \times \mathbb{R}^{\widetilde{E}} \\ a & \longmapsto & a^* \end{pmatrix} \text{ where } \begin{cases} \forall i \in V,\ a^*_i = \begin{cases} a_i \text{ if } i \in V \\ 0 \text{ otherwise} \end{cases} \\ \\ \forall e \in E,\ a^*_e = \begin{cases} a_e \text{ if } e \in E \\ 0 \text{ otherwise} \end{cases} \end{cases}$$

To stick to the Property 3.10's framework, we could set additionally $f^\pi = 0$.

• *Lifting, or transposing using φ^π*

The following lifting theorem is exactly the Property 3.10*ii* where function ψ, φ, and f have been instantiated as proposed above.

Property 3.17 (*Padberg [31]*)

> ***If*** *$a \cdot z \leqslant \alpha$ defines a facet of QP^n* ***then*** *$\varphi^\pi(a) \cdot z \leqslant \alpha$ defines a facet of QP^N.*

3.4.6 Application to QP^n and $P^n_{\delta,X}$

It is important to notice that any valid (resp. facet defining) inequality for CUT^n is also valid (resp. facet defining) for $P^n_{\delta,X}$, but the converse is not true, since $P^n_{\delta,X}$ is in bijection with CUT^{n+1}, not with CUT^n.

- *Identifying the affine map ψ*

$$\boldsymbol{\psi} = \begin{pmatrix} \mathbb{R}^V\times\mathbb{R}^E & \longrightarrow & \mathbb{R}^V\times\mathbb{R}^E \\ (\delta, X) & \longmapsto & (\delta, y) \end{pmatrix} \text{ where } \forall (i,j)\in E,\ y_{i,j} = \frac{\delta_i+\delta_j - X_{i,j}}{2}$$

$$\boldsymbol{\psi^{-1}} = \begin{pmatrix} \mathbb{R}^V\times\mathbb{R}^E & \longrightarrow & \mathbb{R}^V\times\mathbb{R}^E \\ (x, y) & \longmapsto & (x, X) \end{pmatrix} \text{ where } \forall (i,j)\in E,\ X_{i,j} = x_i + x_j - 2\,y_{i,j}$$

One can check that ψ is a bijective linear map and that $\forall S\subseteq V,\ \psi((\mathbb{I}_S, \mathbb{I}_{S:V\setminus S})) = z^S$.
We deduce that $\psi(S^n_{\delta,X}) = S^n_{x,y}$ and $\boxed{\psi(P^n_{\delta,X}) = QP^n.}$

- *Identifying functions φ and f*

As previously, by rewriting the expression $a\cdot\psi(\delta, X)$ as a weighted sum of variables x and y, we identify the coefficient of a vector b such that $a\cdot\psi(\delta, X) = b\cdot(\delta, X)$. This way we obtain the following definition of φ.

$$\boldsymbol{\varphi} = \begin{pmatrix} \mathbb{R}^V\times\mathbb{R}^E & \longrightarrow & \mathbb{R}^V\times\mathbb{R}^E \\ a & \longmapsto & b \end{pmatrix} \text{ where } \begin{cases} \forall i\in V,\ b_i = a_i + \frac{1}{2}\,a\Big(\{i\} : \widetilde{V}\setminus\{i\}\Big) \\ \forall e\in E,\ b_e = -\frac{1}{2}\,a_e \end{cases}$$

By construction of φ, we have $\forall a\in\mathbb{R}^V\times\mathbb{R}^E, \forall(\delta, X)\in\mathbb{R}^V\times\mathbb{R}^E,\ a\cdot\psi(z) = \varphi(a)\cdot(\delta, X)$.
To stick to the Property 3.10's framework, we could set additionally $f = 0$.

- *Transposing inequalities using φ and f*

Transposition 3.18 (*Trivial inequalities*)

Let $(i,j)\in V^<$.
We have $(3.1)_{i,j} \xrightarrow{\varphi,0} (X.4)_{i,j}$, $(3.2)_{i,j} \xrightarrow{\varphi,0} (X.2)_{i,j}$, $(3.3)_{i,j} \xrightarrow{\varphi,0} (X.1)_{i,j}$, and $(3.4)_{i,j} \xrightarrow{\varphi,0} (X.3)_{i,j}$.

According to this transposition, inequalities $(X.1-X.4)$ will be called trivial inequalities for $P^n_{\delta,X}$.

Transposition 3.19 (*Triangle inequalities*)

Let $(i,j,k)\in V^{<<}$. We have $(3.1)_{i,j,k} \xrightarrow{\varphi,0} (3.16)_{i,j}$ and $\left.\begin{matrix}(3.2)_{i,j}\\(3.3)_{i,j}\\(3.4)_{i,j}\end{matrix}\right\} \xrightarrow{\varphi,0} \left\{\begin{matrix}(3.13)_{i,j,k}\\(3.14)_{i,j,k}\\(3.15)_{i,j,k}\end{matrix}\right.$

According to this transposition, in the sequel the triangle inequalities for $P^n_{\delta,X}$ will refer to inequalities $(3.13-3.16)$, even if they are also the triangle inequalities for CUT^n.

Transposition 3.20 (*Clique inequalities*)

Let $S\subseteq V$ such that $|S|\geqslant 2$, and $\theta\in[1..|S|-2]$.
We have $(3.9)_{S,\alpha} \xrightarrow{\varphi,0} (3.19)_{S,\theta}$ where inequalities (3.19) are defined as follows.

$$\forall S\subseteq V,\ \forall\theta\in\Big[1\,..\,|S|-2\Big],\ \ (2\theta-|S|+1)\,\delta(S) + X(S^<) \leqslant \theta\,(\theta+1) \tag{3.19}$$

One can note that, when $|S|$ is odd, the inequality $(3.19)_{S,\theta}$ for $\theta = \frac{|S|-1}{2}$ is exactly $(3.18)_S$. As previously for QP^n, the clique inequalities for $P^n_{\delta,X}$ generalize the clique inequalities for CUT^n.

Transposition 3.21 (*Cut inequalities*)

Let $(S,T) \subseteq \mathcal{P}(V)^2$ such that $S \cap T \neq \emptyset, |S| \geqslant 1$, and $|T| \geqslant 2$.

We have $\begin{cases} (3.10)_{S,T} \xrightarrow{\varphi,0} (3.20)_{S,T} \\ (3.11)_{S,T} \xrightarrow{\varphi,0} (3.21)_{S,T}. \end{cases}$ *where inequalities (3.20) and (3.21) are defined as follows.*

$$\forall (S,T) \subseteq \mathcal{P}(V)^2 \text{ s.t. } S \cap T \neq \emptyset,\ s = |S| \geqslant 1,\ t = |T| \geqslant 2,$$
$$(t-s-1)\,(\delta(S) - \delta(T)) + X(S^<) + X(T^<) - X(S:T) \leqslant 0 \tag{3.20}$$

$$\forall (S,T) \subseteq \mathcal{P}(V)^2 \text{ s.t. } S \cap T \neq \emptyset,\ s = |S| \geqslant 1,\ t = |T| \geqslant 2,$$
$$(t-s-1)\,(\delta(T) - \delta(S)) + X(S^<) + X(T^<) - X(S:T) \leqslant (t-s)\,(t-s-1) \tag{3.21}$$

Transposition 3.22 (*Odd sub-cycle inequalities*)

Let $C \in \mathcal{C}y(K_n)$. Let $S = \text{supp}(C)$. Let F be an odd sub-cycle of C, i.e. $F \subseteq C$ s.t. $|F|$ is odd.

We have $(3.12)_{C,F} \xrightarrow{\varphi,0} (3.17)_{C,F}$.

Note that the transposition of inequality $(3.12)_{C,F}$ produces the following inequality, which can be simplified into $(3.17)_{C,F}$ using that $|F|$ is odd and thus $\left\lfloor \frac{|F|}{2} \right\rfloor = \frac{|F|-1}{2}$.

$$-\frac{1}{2}X(C \setminus F) + \frac{1}{2}X(F) \leqslant \left\lfloor \frac{|F|}{2} \right\rfloor$$

In the next section, we present some experimental results for assessing the reinforcement of Formulation F^2 using the triangle and clique inequalities for $P^n_{\delta,X}$.

3.5 Numerical experiments

We provide in this section numerical experiments conducted in the framework as for Section 2.3, *Cf.* page 89. We only solve linear relaxations, which corresponds to the root node of a Branch-and-Bound algorithm to exactly solve the different formulations. For Tables 3.1 and 3.2, entries are the following.

n: the number of tasks
time: the average running time over the ten n-task instances
gap: the average optimality gap[2] over the ten n-task instances,

3.5.1 Reinforcing F^2 using triangle inequalities

Table 3.1 shows that the lower bound provided by the linear relaxation F^2-LP of F^2 is far from the optimal value (see the third column). Note that other experiments show that F^3 provides the same lower bound. We try to strengthen this lower bound by adding CPLEX cuts[3] and/or the triangle inequalities (3.13 – 3.16).

	F^2-LP		F^2-LP + CPLEX Cuts		F^2-LP + Triangle		F^2-LP + Triangle +CPLEX Cuts	
n	time	gap	time	gap	time	gap	time	gap
10	0	41%	3	0%	0	3%	2	0%
20	0	68%	3	0%	1	13%	2	10%
30	0	77%	5	4%	1	19%	12	18%
40	0	83%	10	27%	32	22%	48	21%
50	1	86%	27	42%	177	23%	145	22%
60	1	93%	375	45%	746	24%	337	24%

Table 3.1: Improvement of the lower bound by adding CPLEX cuts and the 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 induces a much better lower bound than adding the CPLEX cuts, and combining both of them provides almost the same bound as the one obtained by adding only the 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.

3.5.2 Reinforcing F^2 using clique inequalities

As shown by the previous experiments, adding $\mathcal{O}(n^3)$ triangle inequalities increases the time for solving F^2-LP. Since the number of clique inequalities is $\mathcal{O}(n\,2^n)$, we decided to add the clique inequalities using a separation algorithm.

In contrast with the non-overlapping inequalities in Formulations F^3 and F^4, the clique inequalities are not necessary in F^2. Indeed, the integrity constraints suffice, and clique inequalities are only used within the Branch-and-Bound algorithm to improve the lower bound obtained at each node of the branching-tree. Therefore, we can use a heuristic separation algorithm. Such an algorithm provides, for a given point $(\delta^*, X^*) \in P^n_{F^2}$, a set of clique inequalities that (δ^*, X^*) violates. Contrary

[2]For a given instance whose optimal value is OPT, the **optimality gap** of a lower bound LB is $(OPT - LB)/OPT$.
[3]CPLEX cuts are linear inequalities generated by CPLEX to cut off fractional points.

to an exact algorithm, the provided inequality set can be empty, even if (δ^*, X^*) violates some clique inequalities. Let us present the heuristic separation algorithm that we use for our experiments.

• *A heuristic separation algorithm for clique inequalities*

Let $(\delta^*, X^*) \in P^n_{F^2}$. To determinate if (δ^*, X^*) violates some clique inequalities, one can proceed by enumeration on parameters θ and $|S|$, that is to compute, for each $\theta \in [1..n-2]$ and each $s \in [\theta+2..n]$, a subset $S \subseteq V$ of size s that maximizes $(2\theta - (s-1))\,\delta^*(S) + X^*(S^<)$. If the value obtained for a maximizer S is larger than $\theta\,(\theta+1)$, then inequality $(3.19)_{S,\theta}$ is violated by (δ^*, X^*), otherwise, no inequality (3.19) associated with a set of size s and θ is violated by (δ^*, X^*).

If the latter test is done, for each pair (s, θ), with a heuristic solution S, this procedure gives a heuristic separation algorithm. We propose below an implementation of such a heuristic separation algorithm, based on a sorting of δ^* values.

For a given set of parameters (s, θ), we provide a heuristic set S using the following ideas. If $s < 2\theta+1$, then $2\,\theta - (s-1) > 0$. Hence, the larger $\delta^*(S)$ is, the larger $(2\theta - (s-1))\,\delta^*(S) + X^*(S^<)$ is. Therefore, we choose for S a subset corresponding to the s largest components of δ^*. On the contrary, if $s > 2\theta+1$, we choose for S a subset corresponding to the s smallest components of δ^*, since $\delta^*(S)$ is multiplied by a negative coefficient. In these two cases, we focus on maximizing the first term of $(2\theta - (s-1))\,\delta^*(S) + X^*(S^<)$. When $s = 2\theta+1$, the first term is zero, then we focus on maximizing $X^*(S^<)$. Therefore, we choose for S a subset corresponding the $\lceil s/2 \rceil$ largest and the $\lfloor s/2 \rfloor$ smallest components of δ^*, in order to have in $S^<$ as many as possible edges between a node with a small value according to δ^* and a node with a large one. Indeed, thanks to inequalities ($X.1$–$X.2$), $X^*_{i,j} \geqslant |\delta^*_i - \delta^*_j|$. Hence, the closer to 0 is δ^*_i and the closer to 1 is δ^*_j (or vice versa), the larger $X^*_{i,j}$ is.

A heuristic separation of clique inequalities (3.19)

---

input: $(\delta^*, X^*) \in \mathbb{R}^V \times \mathbb{R}^E$, $n = |V|$
output: $\mathcal{I}$ a set of pairs (S, θ) such that (δ^*, X^*) violates $(3.19)_{S,\theta}$, potentially empty

---

$\mathcal{I} \leftarrow \emptyset$
$\sigma \leftarrow$ an order on V that sorts δ^* components such that $\delta_{\sigma(1)} \geqslant \delta_{\sigma(2)} \geqslant \ldots \geqslant \delta_{\sigma(n)}$
for $\theta = 1$ to $n-2$
  for $s = 1$ to min$(2\,\theta, n)$
    $S \leftarrow \sigma\big([1..s]\big)$
    if $(2\theta - (s-1))\,\delta^*(S) + X^*(S^<) > \theta\,(\theta+1)$
      $\mathcal{I} \leftarrow \mathcal{I} \cup \{(S, \theta)\}$
  if $2\,\theta+1 \leqslant n$
    $s \leftarrow 2\,\theta+1$, $S \leftarrow \sigma\big([1..\theta]\big) \cup \sigma\big([n-\theta..n]\big)$
    if $(2\theta - (s-1))\,\delta^*(S) + X^*(S^<) > \theta\,(\theta+1)$
      $\mathcal{I} \leftarrow \mathcal{I} \cup \{(S, \theta)\}$
  for $s = 2\,\theta+2$ to n
    $S \leftarrow \sigma\big([n-s+1..n]\big)$
    if $(2\theta - (s-1))\,\delta^*(S) + X^*(S^<) > \theta\,(\theta+1)$
      $\mathcal{I} \leftarrow \mathcal{I} \cup \{(S, \theta)\}$
return $\mathcal{I}$

Table 3.2 shows that adding the clique inequalities[4] to F^2-LP provides a better lower bound than adding the CPLEX cuts. For example, it reduces the average for 60-task instances from 93% to 27%, while adding CPLEX cuts only reduces it to 45%. In contrast with the triangles inequalities, no time is saved by combining the CPLEX cuts to the clique inequalities, and the gap is only slightly reduced. Moreover, one can note that the solving time with the clique inequalities is much smaller than the solving time with the triangle inequalities. However, this time is again 15 times larger than the solving time for F^2-LP plain. That is too much for an exact solving, where such kind of linear relaxations are solved thousands of times.

	F^2-LP		F^2-LP + CPLEX Cuts		F^2-LP + Clique		F^2-LP + Clique +CPLEX Cuts	
n	time	gap	time	gap	time	gap	time	gap
10	0	41%	3	0%	0	1%	0	0%
20	0	68%	3	0%	0	9%	0	8%
30	0	77%	5	4%	0	14%	1	14%
40	0	83%	10	27%	2	18%	4	17%
50	1	86%	27	42%	6	22%	8	22%
60	1	93%	375	45%	11	27%	15	28%

Table 3.2: Improvement of the lower bound by adding CPLEX cuts and the clique inequalities

In a nutshell, even if they lead to a better relaxation value, triangle inequalities and clique inequalities increase the computation time needed to solve the linear relaxation of Formulation F^2. Therefore, we investigate in the sequel another way of reinforcing Formulation F^2.

[4]Note that the clique inequalities are not all added since we use a *heuristic* separation algorithm to handle them.

Chapter 4

Excluding trivial cuts using facet defining inequalities

4.1 Introduction

In the previous chapter, the polyhedron $P^n_{\delta,X}$ was studied without preferred direction as no objective function was given. We have presented facet defining inequalities that cut off fractional points no matter the value of these points. Conversely, in the sequel, the idea is to reinforce formulation F^2 by cutting off integer points of $P^n_{\delta,X}$ corresponding to solutions which are known to be non-optimal for the objective function. Note that the results obtained for $P^n_{\delta,X}$ in this chapter can be transposed to the cut polytope CUT^{n+1} and to the boolean quadratic polytope QP^n using the methods proposed in Section 3.4.

More precisely, in this chapter, we provide inequalities cutting the point $0 \in \mathbb{R}^V \times \mathbb{R}^E$ since it corresponds to a non-optimal solution. Indeed, this point represents a schedule without any early task, in particular without an on-time task, and the d-schedules are strictly dominant (*Cf.* Lemma 0.1).

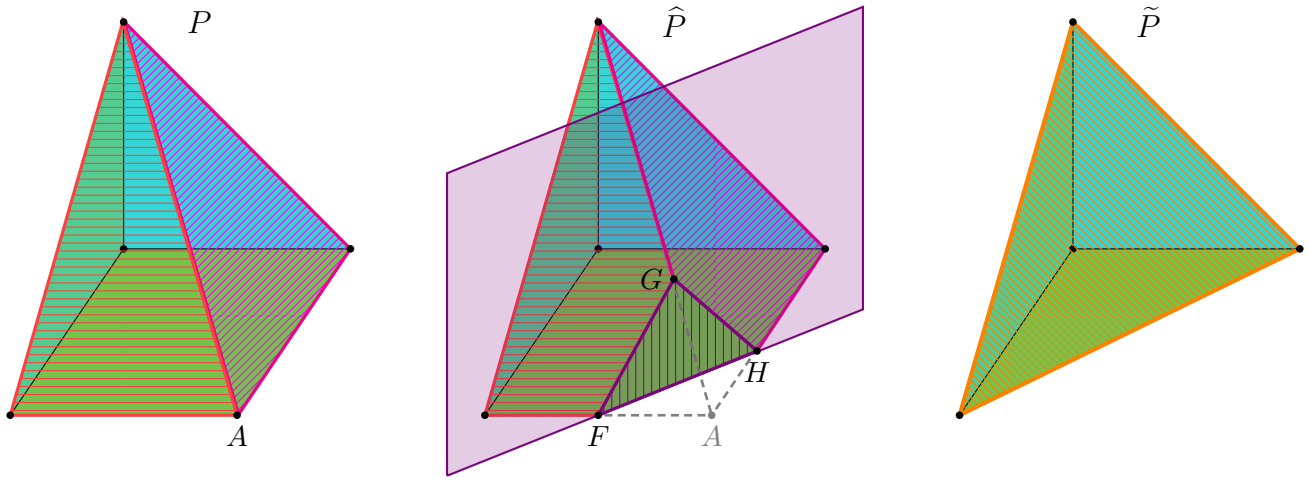


Figure 4.1: A polyhedron included in the hyper-cube,
and two different ways to cut off one of its extreme points

As you can observe in Figure 4.1, an arbitrary inequality cutting off an extreme point A of a polyhedron P included in the hyper-cube can introduce new extreme points (*Cf.* vertices F, G, H of $\widehat{P}$), which are necessarily fractional points. To avoid the occurrence of fractional extreme points, the point to remove must be cut off by inequalities that define facets of the convex hull $\widetilde{P}$ of the other extreme points. (*Cf.* the front facet of $\widehat{P}$ shown in orange in Figure 4.1). Therefore, even if the

inequality $\delta(V)\geqslant 1$ suffices to cut off $0\in\mathbb{R}^V\times\mathbb{R}^E$, we will study the facets of the polyhedron obtained as convex hull of the other points. We then introduce $\widetilde{\boldsymbol{P}}^{\boldsymbol{n}}_{\boldsymbol{\delta,X}}=\operatorname{conv}\widetilde{S}^n_{\delta,X}$ where $\widetilde{\boldsymbol{S}}^{\boldsymbol{n}}_{\boldsymbol{\delta,X}}=S^n_{\delta,X}\backslash\{0\}$. Since $\widetilde{S}^n_{\delta,X}\subseteq S^n_{\delta,X}$, we can describe a point of $\widetilde{S}^n_{\delta,X}$ by only giving its first n components, Lemma 0.5 shows how X can be deduced from δ.

We cannot say in general if removing a point of a set reduces the dimension of its convex hull, even if the removed point is an extreme point of the convex hull. The following property states that removing 0 from $S^n_{\delta,X}$ does not reduce its dimension.

Property 4.1

The polyhedron $\widetilde{P}^n_{\delta,X}$ is fully dimensional, i.e. $\dim(\widetilde{P}^n_{\delta,X})=\dim(\mathbb{R}^V\times\mathbb{R}^E)=\frac{n(n+1)}{2}$.

Proof: Since $\widetilde{P}^n_{\delta,X}=\operatorname{conv}\widetilde{S}^n_{\delta,X}$, it suffices to show that $\widetilde{S}^n_{\delta,X}$ generates $\mathbb{R}^V\times\mathbb{R}^E$, *i.e.* $\operatorname{aff}(\widetilde{S}^n_{\delta,X})=\mathbb{R}^V\times\mathbb{R}^E$.

For any $U\subseteq V$, let us denote by χ^U the point $(\delta,X)\in S^n_{\delta,X}$ such that $\delta_u=1$ if and only if $u\in U$, *i.e.* $\chi^U=(\mathbb{I}_U\in\mathbb{R}^V,\mathbb{I}_{U:V\backslash U}\in\mathbb{R}^E)$. Note that $\chi^U\in\widetilde{S}^n_{\delta,X}$ if $U\neq\emptyset$.

Let us denote by $(e_u)_{u\in V}$ (resp. by $(e_{u,v})_{(u,v)\in V^<}$) the canonical basis of $\mathbb{R}^V$ (resp. of $\mathbb{R}^E$), and by 0_V (resp. 0_E) the zero of $\mathbb{R}^V$ (resp. $\mathbb{R}^E$).

The canonical basis of $\mathbb{R}^V\times\mathbb{R}^E$ can then be written as $\Big((e_u,0_E)_{u\in V}\,,\,(0_V,e_{u,v})_{(u,v)\in V^<}\Big)$.

For any $u\in V$, we have $(e_u,0_E)=\frac{1}{2}(\chi^{\{u\}}-\chi^W+\chi^V)$ where $W=V\backslash\{u\}$.

For any $(u,v)\in V^<$, we have $(0_V,e_{u,v})=\frac{1}{2}(\chi^{\{u\}}+\chi^{\{v\}}-\chi^W)$ where $W=V\backslash\{u\}$.

We deduce that the canonical basis of $\mathbb{R}^V\times\mathbb{R}^E$ can be obtained by linear combination of points of $\widetilde{S}^n_{\delta,X}$.

We then have $\mathbb{R}^V\times\mathbb{R}^E\subseteq\operatorname{vect}(\widetilde{S}^n_{\delta,X})\subseteq\operatorname{vect}(\widetilde{P}^n_{\delta,X})\subseteq\mathbb{R}^V\times\mathbb{R}^E$, and thus $\operatorname{vect}(\widetilde{P}^n_{\delta,X})\subseteq\mathbb{R}^V\times\mathbb{R}^E$. □

• *Link with the non-trivial cut polytope*

A cut is said **trivial** if it does not contain any edge, that is if it corresponds to the bi-partition of nodes $\{\emptyset,V\}$. Using only edge variables X to encode cuts, there is only one point encoding a trivial cut: 0_E. Conversely, using δ,X variables, two points encode a trivial cut: $0=(0_V,0_E)$ and $(\mathbb{I}_V,0_E)$. Let us introduce $\widetilde{\widetilde{P}}^n_{\delta,X}$, the polytope of the non-trivial cuts encoded by δ,X variables, which is formally defined as follows.

$$\widetilde{\widetilde{\boldsymbol{P}}}^{\boldsymbol{n}}_{\boldsymbol{\delta,X}}=\operatorname{conv}\Big(S^n_{\delta,X}\backslash\{(0_V,0_E),(\mathbb{I}_V,0_E)\}\Big)$$

We choose to cut off $(0_V,0_E)$ and not $(\mathbb{I}_V,0_E)$, since $(0_V,0_E)$ is non-optimal for the objective function coming from the UCDDP, while $(\mathbb{I}_V,0_E)$, on the contrary, can be the unique optimal solution for some (very particular) instances. However, from a polyhedral point of view, cutting off $(0_V,0_E)$ or $(\mathbb{I}_V,0_E)$ is equivalent, since these two points are symmetric for $\psi^V_{\delta,X}$. As explained in Section 3.4.4, a facet defining inequality of $\widetilde{\widetilde{P}}^n_{\delta,X}$ that cuts off $(0_V,0_E)$ can be translated into a facet defining inequality that cuts off $(\mathbb{I}_V,0_E)$.

• *First glimpse on $\widetilde{P}^n_{\delta,X}$ using PORTA*

To study this polyhedron for an arbitrary n, we start by studying it for small values of n using a polyhedron representation transformation software, which is in particular able to provide the inequalities defining the facets of $\operatorname{conv}(S)$ for a set of small dimensional points S. We use the software PORTA [2] to this end. We launch PORTA on $\widetilde{S}^n_{\delta,X}$ for $n=4$ (resp. $n=5$), that is on a set of 10-dimensional (resp. 15-dimensional) points, we obtain 68 (resp. 693) inequalities describing $P^4_{F^2}$ (resp. $P^5_{F^2}$).

Roughly speaking, $\widetilde{P}^n_{\delta,X}$ with respect to $P^n_{\delta,X}$ is the same as $\widetilde{P}$ with respect to P in Figure 4.1:

- The initial and the new polyhedra have the same dimension, *i.e.* $\dim(P) = \dim(\widetilde{P})$
- Some facets of the initial polyhedron (*i.e.* P), like the back facet of P, are still facet of the new one (*i.e.* $\widetilde{P}$), or at least the corresponding inequality is still facet defining, even if the facet have smaller size, (*see* te bottom facet of P for example).
- Some facets of the initial polyhedron, like the two striped faced at the front of P, are no longer facet of the new polyhedron, even if they are necessarily still valid.
- Some facets of the new polyhedron, like the orange striped front facet of $\widetilde{P}$, are new in the sense that they are not facet-defining and even not valid for the initial polyhedron. These facets are the facets that we look for, the facets eliminating the point without introducing fractional extreme points.

However, a remarkable difference between $\widetilde{P}$ and $\widetilde{P}^n_{\delta,X}$ in the Figure 4.1 is the number of new facets appearing in the new polyhedron. In the figure, $\widetilde{P}$ has only one new facet, while $\widetilde{P}^n_{\delta,X}$ presents 22 new facets for $n=4$ and at least 325 for $n=5$.

As for $P^n_{\delta,X}$, we can show that the trivial inequalities, the triangle inequalities, the clique inequalities and the generalized cut inequalities define facet of $\widetilde{P}^n_{\delta,X}$ for any $n \in \mathbb{N}$. Among the other inequalities of $\widetilde{P}^n_{\delta,X}$ (*i.e.* the new inequalities, not inherited from $P^n_{\delta,X}$) provided by PORTA for $n=4$ and $n=5$, we identify five families of linear inequalities which can be generalized for any $n \geqslant 4$. Except for the last family, these inequalities defines facets of $\widetilde{P}^n_{\delta,X}$ for any $n \geqslant 4$. The sequel of this chapter presents these inequalities, more precisely each section consists in defining a family and then prove that any inequality of this family defines a facet of $\widetilde{P}^n_{\delta,X}$.

• $P^4_{F^2}$ *case*

PORTA provides 68 inequalities describing $P^4_{F^2}$. They are distributed as follows[1] .

+ inequalities inherited from $P^n_{\delta,X}$:
  - 24 trivial inequalities, namely $(X.1 - X.4)$ for each $(i,j) \in V^<$
  - 16 triangles inequalities, namely $(3.13 - 3.16)$ for each $(i,j,k) \in V^{<<}$
  - 4 generalized cut inequalities, namely (3.21) associated with $(\{i\}, V \setminus \{i\})$ for each $i \in V$
  - 2 clique inequalities, namely (3.19), associated with $S = V$ and $\theta \in \{1,2\}$
+ new inequalities:
  - 12 Hamiltonian path inequalities, *i.e.* $(\mathrm{H}_{\mathcal{P},1,n})$ for $\mathcal{C}$ corresponding to a permutation of V
  - 6 "without name" inequalities, *i.e.* $(?_{u,v})$ for each $(u,v) \in V^<$
  - 4 star inequalities, *i.e.* (Star_u) for each $u \in V$

[1]Since the introduced families of valid inequalities are not disjoint, this distribution could be presented differently: for example some triangle inequalities could have been presented as cut, generalized cut, or clique inequalities.

4.2 How to prove that inequalities define facets

The proofs for the five inequality families follow the same scheme, which is classical in polyhedral studies (*Cf.* [11], page 106). We propose here to recall the scheme of these proofs in order to give to the reader some guidelines in these quite long proofs. However, each proof is written *in extenso* and can be read independently.

Sketch of proof
For a given $n \in \mathbb{N} \setminus [0..2]$, to prove that the following inequality (I) defines a facet of $\widetilde{P}^n_{\delta,X}$.

$$b \cdot (\delta, X) \leqslant \beta \qquad (I)$$

• *Validity*
The first step is to prove that (I) is valid for the polyhedron $\widetilde{P}^n_{\delta,X}$. Since this polyhedron is defined as the convex hull of $\widetilde{S}^n_{\delta,X}$, it suffices to show that (I) is valid for the points of $\widetilde{S}^n_{\delta,X}$.
We consider different cases for (δ, X) in $\widetilde{S}^n_{\delta,X}$ according to the value of the δ components appearing in (I), *i.e.* having non-zero coefficients in (I).

• *Integer points satisfying inequality (I) to equality*
The second step is to identify the points of $\widetilde{S}^n_{\delta,X}$ which satisfy (I) to equality. For this identification, we use the δ case analysis done at the first step. Establishing that some points of $\widetilde{S}^n_{\delta,X}$ satisfy (I) to equality proves that (I) defines a non-empty face of $\widetilde{P}^n_{\delta,X}$, since that shows that some points of $\widetilde{P}^n_{\delta,X}$ are in the hyperplane defined by (I). Let us denote by F this face. In the sequel, we will only say face for non-empty face.

• *General reasoning on facets*
To prove that a face F is a facet, we have to show that its dimension is equal to $d-1$, where d is the dimension of the polyhedron. In case of a polyhedron P defined as the convex hull of a set of initial points S, *i.e.* $P = \mathrm{conv}(S)$, it suffices to consider the dimension of the space generated by the initial points included in the face, *i.e.* $\mathrm{aff}(F \cap S)$. Indeed, the affine space generated by a convex set is also generated by its extreme points. Therefore, the dimension of a face is the dimension given by its extreme points, which are necessarily initial points for such a polyhedron, *i.e.* $\dim(F) = \dim(\mathrm{extr}(F)) = \dim(F \cap S)$.
When the polyhedron is fully dimensional, a way to prove that $\dim(F) = d$ is thus to provide a family of d initial points included in the face, and to show that they are affinely independent. We follow a different line, which is actually equivalent to provide a family of initial points included in the face and show that the rank of this family is $d-1$, since its orthogonal complement is 1-dimensional. In particular, we consider more than d points and do not show they are independent.

• *Defining the including facet*
Since a face of a polyhedron is always included in a facet, we consider a facet F^a containing F. The last (but not least) step, consists in showing that this facet F^a is exactly the face F.
More precisely, we introduce a vector $a \in \mathbb{R}^V \times \mathbb{R}^E$ and a real number $\alpha \in \mathbb{R}$ such that F^a is defined by the inequality $a \cdot (\delta, X) \leqslant \alpha$. We then use initial points included in F, *i.e.* $F \cap S$ to derive necessary conditions on a. Indeed, a point (δ, X) included in F is necessary included in F^a, and thus we have $a \cdot (\delta, X) = \alpha$. Progressively, we deduce from these equalities the values of a components (*Cf.* framed results in the proof). The goal is to deduce that $a = \lambda\, b$ and $\alpha = \lambda\, \beta$ for a certain $\lambda \in \mathbb{R}$, which proves that $F = F^a$, and thus that F is a facet. Note that this goal is reached if and only if the rank of the points considered to derive conditions on a is $\dim(\mathbb{R}^V \times \mathbb{R}^E) - 1 = \dim(\widetilde{P}^n_{\delta,X}) - 1$.

In spite of their common structure, the following proofs do not seem to be able to be gathered into a common meta-proof. Indeed, the integer points satisfying inequalities to equality and thus the third step of proofs have different nature. For example, the third step of the proof for the Hamiltonian path inequalities requires to prove one intermediate property by induction on the number of vertices on the path between the two endpoints of an edge, while the proof for the Hamiltonian cycle inequalities requires in addition an induction on the vertices along the cycle. Conversely, the other proofs do not require any induction reasoning.

4.3 Hamiltonian path inequalities

For any Hamiltonian path $\mathcal{P}$ of K_n, we introduce the following inequality where u and v denote the endpoints of $\mathcal{P}$.

$$\delta_u + \delta_v + X(\mathcal{P}) \geqslant 2 \qquad (\mathrm{H}_{\mathcal{P}})$$

Property 4.2

For any Hamiltonian path $\mathcal{P}$ the inequality $(\mathrm{H}_{\mathcal{P}})$ defines a facet of $\widetilde{P}^n_{\delta,X}$.

Proof: Let $\mathcal{P}$ be a Hamiltonian path of K_n. Since vertices of V can be interchanged without changing K_n, we assume without loss of generality that $\mathcal{P}$ goes from 1 to n, and that edges of $\mathcal{P}$ are the edges $\{i, i+1\}$ for $i \in [1..n[$. The inequality $(\mathrm{H}_{\mathcal{P}})$ can then be written as follows, and Figure 4.2 gives an illustration of the variables involved in this inequality, that is having a non zero coefficient.

$$\delta_1 + \delta_n + \sum_{i=1}^{n-1} X_{i,i+1} \geqslant 2 \qquad (\mathrm{H}_{\mathcal{P},1,n})$$

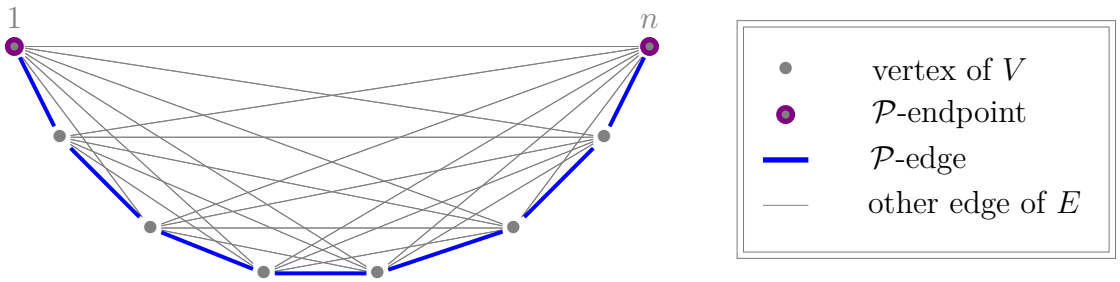


Figure 4.2: Illustration of the terms (δ, X) involved in $(\mathrm{H}_{\mathcal{P},1,n})$

• *Validity*
Let $(\delta, X) \in \widetilde{S}^n_{\delta,X}$.

- If $(\delta_1, \delta_n) = (1,1)$, then $\delta_1 + \delta_n = 2$. As X has only non-negative values, (δ, X) satisfies $(\mathrm{H}_{\mathcal{P},1,n})$.
- If $(\delta_1, \delta_n) = (1,0)$, then there exists (at least) one index $k \in [1..n-1]$ such that $\delta_k = 1$ and $\delta_{k+1} = 0$, and hence such that $X_{k,k+1} = 1$. We deduce that (δ, X) satisfies $(\mathrm{H}_{\mathcal{P},1,n})$.
- If $(\delta_1, \delta_n) = (0,1)$, the situation is symmetric to the previous one. Similarly, (δ, X) satisfies $(\mathrm{H}_{\mathcal{P},1,n})$.
- If $(\delta_1, \delta_n) = (0,0)$, there exists (at least) one index $j \in [2..n-1]$ such that $\delta_j = 1$ since $\delta \neq 0$. Then, there exists one index $k_1 \in [1..j-1]$ such that $\delta_{k_1} = 0$ and $\delta_{k_1+1} = 1$, and another index $k_2 \in [j+1..n-1]$ such that $\delta_{k_2} = 1$ and $\delta_{k_2+1} = 0$. We then have $X_{k_1,k_1+1} = X_{k_2,k_2+1} = 1$ with $k_1 \neq k_2$. Therefore, $\sum_{i=1}^{n-1} X_{i,i+1} \geqslant 2$ and (δ, X) satisfies $(\mathrm{H}_{\mathcal{P},1,n})$.

We deduce that all the points of $\widetilde{S}^n_{\delta,X}$, and hence of $\widetilde{P}^n_{\delta,X} = \mathrm{conv}(\widetilde{S}^n_{\delta,X})$, satisfy inequality $(\mathrm{H}_{\mathcal{P},1,n})$.

• *Introducing the including facet*
Let us consider $a \in \mathbb{R}^V \times \mathbb{R}^E$ and $\alpha \in \mathbb{R}$ such that $F^a = \{(\delta, X) \in \widetilde{P}^n_{\delta,X} \mid a \cdot (\delta, X) = \alpha\}$ is a facet of $\widetilde{P}^n_{\delta,X}$ containing the face F associated with $(\mathrm{H}_{\mathcal{P},1,n})$, that is the face defined as follows.

$$F = \left\{ (\delta, X) \in \widetilde{P}^n_{\delta,X} \;\middle|\; \delta_1 + \delta_n + \sum_{i=1}^{n-1} X_{i,i+1} = 2 \right\}$$

• *Integer points satisfying inequality* $(\mathrm{H}_{\mathcal{P},1,n})$ *to equality*

Let us denote by $p^{K,L}$ the point of $\mathcal{S}$ defined by $\delta = \mathbb{I}_{[K..L]}$, for $(K,L) \in V^{\leqslant}$. Such a point satisfies $(\mathrm{H}_{\mathcal{P},1,n})$ to equality. Actually, the set of points of $\widetilde{S}^n_{\delta,X}$ satisfying $(\mathrm{H}_{\mathcal{P},1,n})$ to equality is exactly $\left\{p^{K,L} \,\middle|\, (K,L) \in V^{\leqslant}\right\}$. Note that these points can be partitioned into seven seven types regarding to 1 and n:

(a) $1 < K = L < n$ (c) $1 < K < L = n$ (e) $1 < K < L = n$ (g) $1 = K < L = n$

(b) $1 = K < L < n$ (d) $1 = K < L < n$ (f) $1 < K < L < n$

Figure 4.3 gives an illustration of the non-zero components of such points.

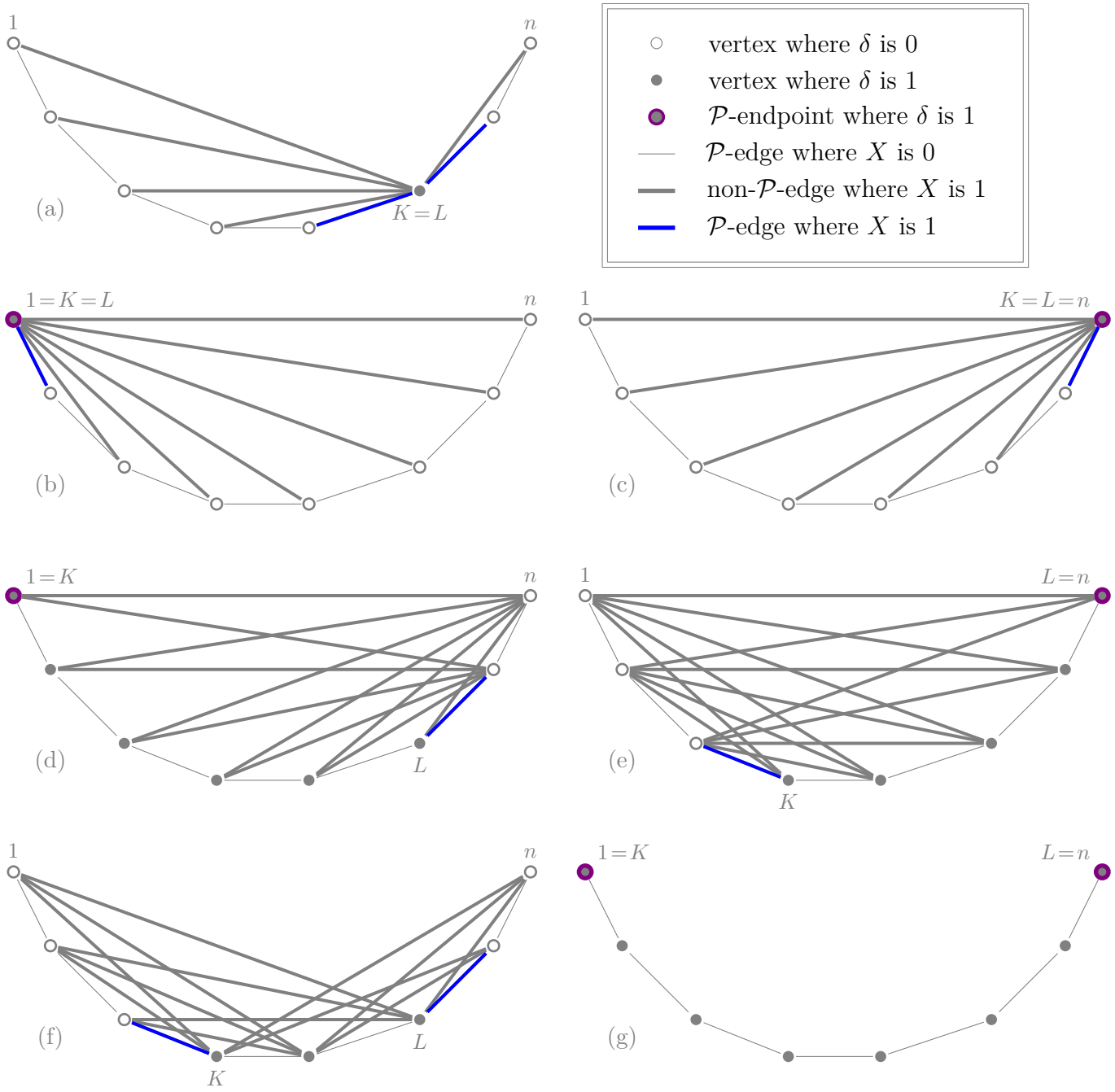


Figure 4.3: Illustration of non zero components of $p^{K,L}$, for the seven different types of (K,L) regarding to 1 and n

For any $(K,L)\in V^{\leqslant}$, $p^{K,L}\in F^a$ since $p^{K,L}\in F$, and we have $a.p^{K,L}=\alpha$, which can be written as follows.

$$\sum_{j=K}^{L} a_j + \sum_{i=1}^{K-1}\sum_{j=K}^{L} a_{i,j} + \sum_{i=K}^{L}\sum_{j=L+1}^{n} a_{i,j} = \alpha \qquad (1_{K,L})$$

Using colors, Figure 4.4 illustrates the non-zero coefficients appearing in this equation $(1_{K,L})$ in case (f), *i.e.* for (K,L) such that $1<K<L<n$.

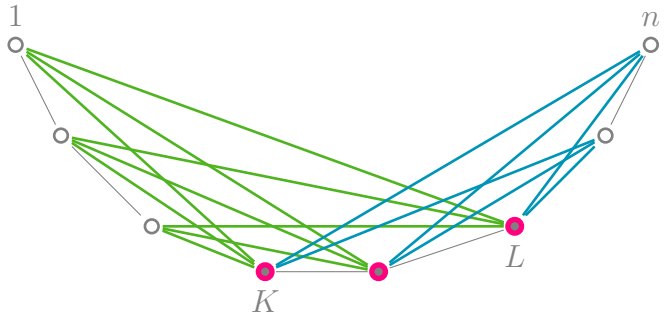


Figure 4.4: Illustration of terms of a involved in $(1_{K,L})$ for (K,L) of type (f)

We now start to use equations $(1_{K,L})$ *for different values of* K *and* L*, which will allow to first prove that* $a_{/V}=\frac{\alpha}{2}(\mathbb{I}_1+\mathbb{I}_n)$.

– With $(K,L)=(1,n)$ the equation $(1_{K,L})$ is simply the following equation.

$$\sum_{j=1}^{n} a_j = \alpha \qquad (1_{1,n})$$

– For any $J\in[1..n-1]$, the equation $(1_{K,L})$ for $(K,L)=(1,J)$ is:

$$\sum_{j=1}^{J} a_j + \underline{0} + \sum_{i=1}^{J}\sum_{j=J+1}^{n} a_{i,j} = \alpha \qquad (1_{1,J})$$

while the equation $(1_{K,L})$ with $(K,L)=(J+1,n)$ is

$$\sum_{j=J+1}^{n} a_j + \sum_{i=1}^{J}\sum_{j=J+1}^{n} a_{i,j} + \underline{0} = \alpha \qquad (1_{J+1,n})$$

Summing these both equations, we obtain:

$$\sum_{j=1}^{n} a_j + 2\sum_{i=1}^{J}\sum_{j=J+1}^{n} a_{i,j} = 2\alpha$$

and using $(1_{1,n})$ to simplify α, we finally obtain:

$$\sum_{i=1}^{J}\sum_{j=J+1}^{n} a_{i,j} = \frac{\alpha}{2} \qquad (\bigstar)$$

– For any $K\in[2..n]$, the equation $(1_{K-1,n})$, can be written as follows,

$$\sum_{j=K-1}^{n} a_j + \sum_{i=1}^{K-2}\sum_{j=K-1}^{n} a_{i,j} + 0 = \alpha \quad \text{that is} \quad \sum_{j=K-1}^{n} a_j + \left(\sum_{i=1}^{K-2}\sum_{j=K}^{n} a_{i,j} + \sum_{i=1}^{K-2} a_{i,K-1}\right) = \alpha \quad (1_{K-1,n})$$

while the equation $(1_{K,n})$ can be written as follows:

$$\sum_{j=K}^{n} a_j + \sum_{i=1}^{K-1}\sum_{j=K}^{n} a_{i,j} + 0 = \alpha \quad \text{that is} \quad \sum_{j=K}^{n} a_j + \left(\sum_{i=1}^{K-2}\sum_{j=K}^{n} a_{i,j} + \sum_{j=K}^{n} a_{K-1,j}\right) = \alpha \qquad (1_{K,n})$$

We deduce that left members of these both inequality are equal, and simplifying their common terms we obtain for any $K\in[2..n]$:

$$a_{K-1} + \sum_{i=1}^{K-2} a_{i,K-1} = \sum_{j=K}^{n} a_{K-1,j} \qquad (2_{K-1})$$

– For any $L\in[1..n-1]$, the equation $(1_{1,L+1})$ can be written as follows:

$$\sum_{j=1}^{L+1} a_j + 0 + \sum_{i=1}^{L+1}\sum_{j=L+2}^{n} a_{i,j} = \alpha \quad \text{that is} \quad \sum_{j=K}^{L+1} a_j + \left(\sum_{i=K}^{L}\sum_{j=L+2}^{n} a_{i,j} + \sum_{j=L+2}^{n} a_{L+1,j}\right) = \alpha \qquad (1_{1,L+1})$$

while the equation $(1_{1,L})$ can be written as follows:

$$\sum_{j=1}^{L} a_j + 0 + \sum_{i=1}^{L}\sum_{j=L+1}^{n} a_{i,j} = \alpha \quad \text{that is} \quad \sum_{j=1}^{L} a_j + \left(\sum_{i=1}^{L}\sum_{j=L+2}^{n} a_{i,j} + \sum_{i=1}^{L} a_{i,L+1}\right) = \alpha \qquad (1_{1,L})$$

We deduce that left members of these both inequality are equal, and simplifying their common terms we obtain for any $L\in[1..n-1]$:

$$a_{L+1} + \sum_{j=L+2}^{n} a_{L+1,j} = \sum_{i=1}^{L} a_{i,L+1} \qquad (3_{L+1})$$

– For $J\in[2..n-1]$, we then have $\sum_{j=J+1}^{n} a_{J,j} = a_J + \sum_{i=1}^{J-1} a_{i,J}$ by (2_J), and $a_J + \sum_{j=J+1}^{n} a_{J,j} = \sum_{i=1}^{J-1} a_{i,J}$ by (3_J), which implies that $a_J = 0$.

$$\boxed{\forall J\in[2..n-1],\ a_J = 0}$$

The equation $(1_{1,n})$ is then equivalent to $a_1 + a_n = \alpha$. Moreover, using $(\bigstar)$ the equation $(1_{1,J})$ can be simplified in $a_1 + \frac{\alpha}{2} = \alpha$. We deduce that $a_n = \dfrac{\alpha}{2}$ and finally we obtain:

$$\boxed{a_1 = a_n = \frac{\alpha}{2}}$$

We continue using equations $(1_{K,L})$ for different values of K and L, which will allow to prove that $a_{/E} = \frac{\alpha}{2}\mathbb{I}_{\mathcal{P}} \in \mathbb{R}^E$, i.e. $\forall i\in[1..n[,\ a_{i,i+1} = \frac{\alpha}{2}$ and $a_e = 0$ for $e\in E\setminus\mathcal{P}$.

For any $k\in[1..n]$, by analogy with inner and outer degrees in oriented graph, we introduce the following notations.

$$\boldsymbol{a}^-(k) = \sum_{i=1}^{k-1} a_{i,k} \quad \text{and} \quad \boldsymbol{a}^+(k) = \sum_{j=k+1}^{n} a_{k,j}$$

– Using this notation, the equation $(1_{K,L})$ for $(K,L)=(J,J)$ is equivalent to $a_J + a^-(J) + a^+(J) = \alpha$.
For $J=1$, $a_J = \frac{\alpha}{2}$ and $a^-(J)=0$, we deduce that $a^+(1)=\frac{\alpha}{2}$. Similarly, we also have $a^-(n)=\frac{\alpha}{2}$.
For any $J\in[2..n-1]$, $a_J=0$, thus the equation $(1_{K,L})$ for $(K,L)=(J,J)$ can the written as follows;

$$a^-(J) + a^+(J) = \alpha \qquad (1_{J,J})$$

– With $(K,L)=(1,2)$ the equation $(1_{K,L})$ can be written as follows.

$$(a_1+a_2) + (a^+(1)-a_{1,2}) + a^+(2) = \alpha \quad \text{that is} \quad \left(\frac{\alpha}{2}+0\right) + +(\frac{\alpha}{2}-a_{1,2}) + a^+(2) = \alpha$$

We deduce that $a^+(2)=a_{1,2}$. Besides, by definition, $a^-(2)=a_{1,2}$, then the equation $(1_{J,J})$ for $J=2$ gives $a_{1,2}+a_{1,2}=\alpha$. We finally obtain:

$$\boxed{a_{1,2}=\frac{\alpha}{2}}$$

and similarly:

$$\boxed{a_{n-1,n}=\frac{\alpha}{2}}$$

– Let $J\in[2..n-2]$. The equation $(1_{K,L})$ for $(K,L)=(1,J+1)$ gives:

$$(a_1+0) + \sum_{i=1}^{J+1}\sum_{j=J+2}^{n} a_{i,j} = \alpha \qquad (1_{1,J+1})$$

while for $(K,L)=(J,n)$ this equation gives:

$$(0+a_n) + \sum_{i=1}^{J-1}\sum_{j=J}^{n} a_{i,j} = \alpha \qquad (1_{J,n})$$

Using that $a_1=a_n$, we deduce that the the two double sums are equal, and reformulate this equality as follows.

$$\sum_{i=1}^{J+1}\sum_{j=J+2}^{n} a_{i,j} = \sum_{i=1}^{J-1}\sum_{j=J}^{n} a_{i,j} \Leftrightarrow \cancel{\sum_{i=1}^{J-1}\sum_{j=J+2}^{n} a_{i,j}} + \sum_{j=J+2}^{n}(a_{J,j}+a_{J+1,j}) = \cancel{\sum_{i=1}^{J-1}\sum_{j=J+2}^{n} a_{i,j}} + \sum_{i=1}^{J-1}(a_{i,J}+a_{i,J+1})$$

$$\Leftrightarrow \left(\sum_{j=J+2}^{n} a_{J,j} + \sum_{j=J+2}^{n} a_{J+1,j}\right) + a_{J,J+1} = \left(\sum_{i=1}^{J-1} a_{i,J} + \sum_{i=1}^{J-1} a_{i,J+1}\right) + a_{J,J+1}$$

$$\Leftrightarrow \sum_{j=\mathbf{J+1}}^{n} a_{J,j} + \sum_{j=J+2}^{n} a_{J+1,j} = \sum_{i=1}^{J-1} a_{i,J} + \sum_{i=1}^{\mathbf{J}} a_{i,J+1}$$

$$\Leftrightarrow a^+(J) + a^+(J+1) = a^-(J) + a^-(J+1)$$

By adding $a^-(J)+a^+(J+1)$ on the both sides of the latter equality, we obtain:

$$\underbrace{(a^+(J)+a^-(J))}_{\alpha} + 2a^+(J+1) = 2a^-(J) + \underbrace{(a^-(J+1)+a^+(J+1))}_{\alpha} \quad \text{and then} \quad a^+(J+1)=a^-(J) \quad (4_J^{+-})$$

while by adding $a^+(J)+a^-(J+1)$, we obtain:

$$2a^+(J) + \underbrace{(a^+(J+1)+a^-(J+1))}_{\alpha} = \underbrace{(a^-(J)+a^+(J))}_{\alpha} + 2a^-(J+1) + \quad \text{and then} \quad a^-(J+1)=a^+(J) \quad (4_J^{-+})$$

Starting from $a^+(2)=a^-(2)=a_{1,2}=\dfrac{\alpha}{2}$ and using alternatively equations (4^{+-}_J) and (4^{-+}_J) for each $J\in[2..n-1]$, we obtain:

$$\frac{\alpha}{2}=a^+(2)=a^-(3)=\ldots=\begin{cases}a^-(n-1) \text{ if } n \text{ is even}\\ a^+(n-1) \text{ otherwise}\end{cases}$$

and

$$\frac{\alpha}{2}=a^-(2)=a^+(3)=\ldots=\begin{cases}a^+(n-1) \text{ if } n \text{ is even}\\ a^-(n-1) \text{ otherwise}\end{cases}$$

To sum up, we have:

$$\forall J\in[2..n-1],\ a^+(J)=a^-(J)=\frac{\alpha}{2} \qquad (\clubsuit)$$

– For any $J\in[2..n-2]$, the equation $(1_{K,L})$ for $(K,L)=(J,J+1)$ is:

$$(a_J+a_{J+1})+a^-(J)+(a^+(J)-a_{J,J+1})+(a^-(J+1)-a_{J,J+1})+a^+(J+1)=\alpha$$

and using $(\clubsuit)$, this is equivalent to $0+2\,\alpha-2\,a_{J,J+1}=\alpha$. We deduce that:

$$\boxed{\forall J\in[2..n-2],\ a_{J,J+1}=\frac{\alpha}{2}} \qquad (\spadesuit)$$

Up to now, we have shown that $a_1=a_n=\dfrac{\alpha}{2}$, $\forall J\in[2..n-1], a_J=0$ *and* $\forall J\in[1..n-1],\ a_{J,J+1}=\frac{\alpha}{2}$.
In the rest of the proof, we will show by induction that all the other components of a *are zero.*

For $l\in[2..n-1]$ let us consider the following property.

$$H_l:\forall J\in[1..n-l\,],\ a_{J,J+l}=0$$

Let us now show that H_l is true by induction on $l\in[2..n-1]$.

• *Rewriting* $(1_{J,J+l})$
For any $l\in[2..n-1]$ and $J\in[1..n-l]$, the equation $(1_{K,L})$ for $(K,L)=(J,J+l)$ can be written as follows:

$$\frac{\alpha}{2}\,\mathbb{I}_{J=1 \text{ or } J+l=n}+\left(a^-(J)+\boxed{\sum_{j=J+l+1}^{n}a_{J,j}}\right)+\sum_{k=J+1}^{k=J+l-1}\left(\boxed{\sum_{i=1}^{J-1}a_{i,k}}+\boxed{\sum_{j=J+l+1}^{n}a_{k,j}}\right)+\left(\boxed{\sum_{i=1}^{J-1}a_{i,J+l}}+a^+(J+l)\right)=\alpha \qquad (1_{J,J+l})$$

where $\mathbb{I}_{J=1 \text{ or } J+l=n}$ is 1 if $j=1$ or $J+l=n$ and 0 otherwise. Using $(\clubsuit)$, we simplify the three non-boxed terms as follows.

$$\frac{\alpha}{2}\,\mathbb{I}_{J=1 \text{ or } J+l=n}+a^-(J)+a^+(J+l)=\left\{\begin{array}{ll}\frac{\alpha}{2}+0+\frac{\alpha}{2} & \text{if } J=1\\ \frac{\alpha}{2}+\frac{\alpha}{2}+0 & \text{if } J+l=n\\ 0+\frac{\alpha}{2}+\frac{\alpha}{2} & \text{otherwise}\end{array}\right\}=\alpha$$

We deduce that the sum of boxed terms is 0. Let us rewrite each of these boxed terms using only coefficients $a_{i,j}$ with $(i,j)\in[J..J+l]^<$ and making explicitly appear the coefficient $a_{J,J+l}$. Using the notation a^+, the first boxed term can be expressed as follows.

$$\boxed{\sum_{j=J+l+1}^{n}a_{J,j}}=\sum_{j=J+1}^{n}a_{J,j}-\sum_{j=J+1}^{J+l}a_{J,j}=\underbrace{a^+(J)-a_{J,J+1}}_{=0 \text{ by } (\clubsuit) \text{ and } (\spadesuit)}-\sum_{j=J+2}^{J+l-1}a_{J,j}-a_{J,J+l}$$

Similarly, for any $k\in[J+1\,..\,J+l-1]$ the corresponding second boxed term can be expressed as follows.

$$\boxed{\sum_{j=J+l+1}^{n} a_{k,j}} = \sum_{j=k+1}^{n} a_{k,j} - \sum_{j=k+1}^{J+l} a_{k,j} = \underbrace{a^+(k) - a_{k,k+1}}_{=0 \text{ by } (\clubsuit) \text{ and } (\spadesuit)} - \sum_{j=k+2}^{J+l} a_{k,j}$$

Using the notation a^-, the last boxed term can be expressed as follows.

$$\boxed{\sum_{i=1}^{J-1} a_{i,J+l}} = \sum_{i=1}^{J+l-1} a_{i,J+l} - \sum_{i=J}^{J+l-1} a_{i,J+l} = \underbrace{a^-(J+l) - a_{J+l-1,J+l}}_{=0 \text{ by } (\clubsuit) \text{ and } (\spadesuit)} - \sum_{i=J+1}^{J+l-2} a_{i,J+l} - a_{J,J+l}$$

Similarly, for any $k\in[J+1\,..\,J+l-1]$ the corresponding third boxed term can be expressed as follows.

$$\boxed{\sum_{i=1}^{J-1} a_{i,k}} = \sum_{i=1}^{k-1} a_{i,k} - \sum_{i=J}^{k-1} a_{i,k} = \underbrace{a^-(k) - a_{k-1,k}}_{=0 \text{ par } (\clubsuit) \text{ et } (\spadesuit)} - \sum_{i=J}^{k-2} a_{i,k}$$

Finally, we can write that the sum of the opposite of the boxed terms is 0 as follows.

$$\boxed{\sum_{j=J+2}^{J+l-1} a_{J,j} + a_{J,J+l}} + \sum_{k=J+1}^{k=J+l-1} \left(\boxed{\sum_{j=k+2}^{J+l} a_{k,j}} + \boxed{\sum_{i=J}^{k-2} a_{i,k}} \right) + \boxed{\sum_{i=J+1}^{J+l-2} a_{i,J+l} + a_{J,J+l}} = 0 \qquad (1'_{J,J+l})$$

∘ *Initialization, for* $l=2$.
For $l=2$ and for any $J\in[1\,..\,n-2]$, the sums in the latter equation are empty sums, we then obtain:

$$0 + a_{J,J+2} + (0+0) + 0 + a_{J,J+2} = 0 \qquad (1'_{J,J+2})$$

We deduce that $\forall J\in[1..n-2]$, $a_{J,J+2}=0$. In other words, Property H_2 stands.

∘ *Induction*
Let $l\in[2\,..\,n-2]$. Let us assume that Property H stands up to the rank l. We will show that Property H_{l+1} stands. Let $J\in[1\,..\,n-(l+1)]$. In its rewritten version, the equation $(1_{K,L})$ for $(K,L)=(J,J+l)$ gives:

$$\boxed{\sum_{j=J+2}^{J+l} a_{J,j} + a_{J,J+l+1}} + \sum_{k=J+1}^{k=J+l} \left(\boxed{\sum_{j=k+2}^{J+l+1} a_{k,j}} + \boxed{\sum_{i=J}^{k-2} a_{i,k}} \right) + \boxed{\sum_{i=J+1}^{J+l-1} a_{i,J+l+1} - a_{J,J+l+1}} = 0 \quad (1'_{J,J+l+1})$$

By induction hypothesis, the coefficients $a_{i,j}$ for $j-i\leqslant l$ are zero. Moreover,

- for any $j\in[J+2\,..\,J+l\,]$, we have $j-J\leqslant J+l-J=l$, thus $a_{J,j}=0$.
- for any $k\in[J+1\,..\,J+l\,]$ and $j\in[k+2\,..\,J+l+1\,]$, we have $j-k\leqslant J+l+1-(j+1)=l$, thus $a_{k,j}=0$.
- for any $k\in[J+1\,..\,J+l\,]$ and $i\in[J\,..\,k-2]$, we have $k-i\leqslant J+l-J=l$, thus $a_{i,k}=0$.
- for any $i\in[J+1\,..\,J+l-1\,]$, we have $J+l+1-i\leqslant J+l+1-(J+1)=l$, thus $a_{i,J+l+1}=0$.

We deduce that all the sums in the latter expression are zero, and therefore, that $a_{J,J+l+1}=0$. This is true for any $J\in[1\,..\,n-(l+1)]$, Then Property H_{l+1} stands.

• *Conclusion*
Finally, $a=\dfrac{\alpha}{2}\left(\mathbb{I}_1+\mathbb{I}_n+\sum_{i=1}^{n-1}\mathbb{I}_{(i,i+1)}\right)$. Since F^a defines a facets, $a\neq 0$, and thus $\alpha\neq 0$. Since dividing all the components of a by α does not change F^a, we can assume that $\alpha=2$ without loss of generality. We then obtain $a=\mathbb{I}_1+\mathbb{I}_n+\sum_{i=1}^{n-1}\mathbb{I}_{(i,i+1)}$, thus $F^a=F$ and F defines a facet. □

4.4 Hamiltonian cycle inequalities

In this section, we assume that $n \geqslant 4$. For any Hamiltonian cycle $\mathcal{C}$ of K_n, we introduce the following inequality for u and v two nodes not consecutive on $\mathcal{C}$.

$$\delta_u + \delta_v - X_{u,v} + X(\mathcal{C}) \geqslant 2 \qquad (\text{H'}_{\mathcal{C},u,v})$$

Property 4.3

For any Hamiltonian cycle $\mathcal{C}$ of K_n the inequality $(\text{H'}_{\mathcal{C},u,v})$ defines a facet of $\widetilde{P}^n_{\delta,X}$.

Proof: Let $\mathcal{C}$ be a Hamiltonian cycle of K_n and (u, v) a pair of nodes not consecutive on $\mathcal{C}$. Since nodes of V can be interchanged without changing K_n, we assume without loss of generality that $u = 1$ and that edges of $\mathcal{C}$ are the edge $\{i, i+1\}$ for $i \in [1..n[$ and $n, 1$. We then have $v \in [3..n-1]$.
Let us denote by W the set of vertices different from u and v, and split it according to their position in $\mathcal{C}$ relatively to u and v:

$$W = V \setminus \{u, v\}, \quad W_1 = [2..v-1] \quad \text{and} \quad W_2 = [v+1..n]$$

Considering the orientation on $\mathcal{C}$ given by the natural order on $[1..n]$, W_1 (resp. W_2) is the set of nodes placed between u and v (resp. between v and u) on $\mathcal{C}$.

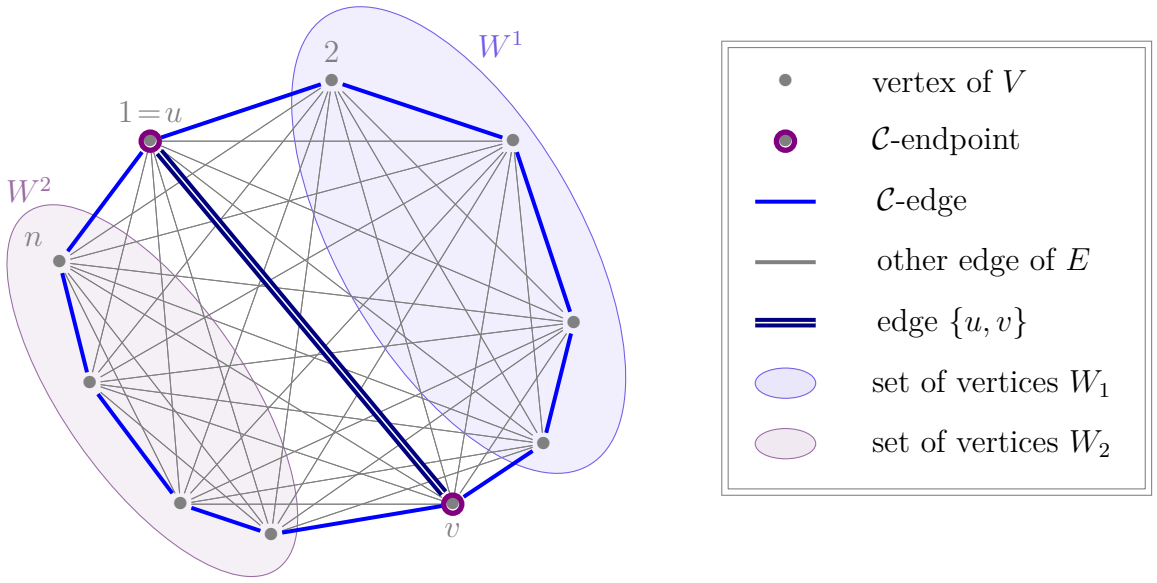


Figure 4.5: Illustration of terms (δ, X) involved in $(\text{H'}_{\mathcal{C},u,v})$

In the sequel, all indices are given modulo n. For example, $\delta_{n+1} = \delta_1 = \delta_u$. That will also be the case for index intervals. Thus, we denote by $[L..K]$ the set $\{L+i \bmod n \,|\, i \in \mathbb{N} \text{ such that } L+i \leqslant K \bmod n\}$. If $(K, L) \in V^{\leqslant}$, this is a standard notation, in particular if $K = L$, we have $[K..L] = \{K\}$. If conversely $K > L$, we have $[L..K] = [L..n] \sqcup [1..K]$.

Moreover, for any $k \in [1..n]$, by analogy with inner and outer degrees in oriented graph, we introduce the following notations.

$$\boldsymbol{a}^-(k) = \sum_{i=1}^{k-1} a_{i,k} \quad \text{and} \quad \boldsymbol{a}^+(k) = \sum_{j=k+1}^{n} a_{k,j}$$

• *Validity*
Let $(\delta, X) \in \widetilde{S}^n_{\delta,X}$.

- If $(\delta_u, \delta_v) = (1,1)$, then $X_{u,v} = 0$ and $\delta_u + \delta_v - X_{u,v} = 2$. Moreover, all X components are non-negative, therefore, $\delta_u + \delta_v - X_{u,v} + X(\mathcal{C}) \geqslant 2$, that is (δ, X) satisfies (H'$_{\mathcal{C},u,v}$).

- If $(\delta_u, \delta_v) = (0,1)$, then $X_{u,v} = 1$ and $\delta_u + \delta_v - X_{u,v} = 0$. Moreover, there exists at least one node $k \in W_1$ such that $\delta_k = 0$ and $\delta_{k+1} = 1$, and another node $l \in W_2$ such that $\delta_l = 1$ and $\delta_{l+1} = 0$. Since $(\delta, X) \in \widetilde{S}^n_{\delta,X}$, we then have $X_{k,k+1} = 1$ and $X_{l,l+1} = 1$. As X components are non-negative we also have $X(\mathcal{C}) \geqslant X_{k,k+1} + X_{l,l+1}$. We deduce that (δ, X) satisfies (H'$_{\mathcal{C},u,v}$).

- If $(\delta_u, \delta_v) = (1,0)$, following the same line we show that (δ, X) satisfies (H'$_{\mathcal{C},u,v}$).

- If $(\delta_u, \delta_v) = (0,0)$, then $X_{u,v} = 0$. Moreover, there exists (at least) one index $w \in [2..n-1]$ such that $\delta_w = 1$ since $\delta \neq 0$. Following the same reasoning for (u,w) as previously for (u,v), we have $X(\mathcal{C}) \geqslant 2$. We deduce that (δ, X) satisfies (H'$_{\mathcal{C},u,v}$).

We deduce that all the points of $\widetilde{S}^n_{\delta,X}$ satisfy inequality (H'$_{\mathcal{C},u,v}$). Since $\widetilde{P}^n_{\delta,X} = \mathrm{conv}(\widetilde{S}^n_{\delta,X})$, we deduce that any point in $\widetilde{P}^n_{\delta,X}$ satisfies also this inequality. In other words, (H'$_{\mathcal{C},u,v}$) is valid for $\widetilde{P}^n_{\delta,X}$.

• *Introducing the including facet*
Let us consider $a \in \mathbb{R}^V \times \mathbb{R}^E$ and $\alpha \in \mathbb{R}$ such that $F^a = \{(\delta, X) \in \widetilde{P}^n_{\delta,X} \mid a \cdot (\delta, X) = \alpha\}$ is a facet of $\widetilde{P}^n_{\delta,X}$ containing the face F associated with (H'$_{\mathcal{C},u,v}$) , that is the face defined as follows.

$$F = \left\{ (\delta, X) \in \widetilde{P}^n_{\delta,X} \;\middle|\; \delta_u + \delta_v - X_{u,v} + X(\mathcal{C}) = 2 \right\}$$

• *Integer points satisfying inequality* (H'$_{\mathcal{C},u,v}$) *to equality*
For any $(K, L) \in V^2$, let us denote by $p^{K,L}$ the point of $\mathcal{S}$ defined by $\delta = \mathbb{I}_{[K..L]}$. Since the index interval is given modulo n, note that the point corresponding to $\delta = \mathbb{I}_V$ can be written $p^{K,L}$ for $(K, L) = (1, n)$ or $(K, L) = (2, 1)$, or $(K, L) = (3, 2)$ and so on. However, in the following, we use preferentially $(K, L) = (1, n)$ to denotes this point.
The point $p^{K,L}$ satisfies (H'$_{\mathcal{C},u,v}$) to equality if and only if $u \notin [K..L]$ or $v \notin [K..L]$ or $[K..L] = [1..n]$.
Indeed, if $u \in [K..L]$ and $v \in [K..L]$, then we have $\underline{\delta_u + \delta_v} - \underline{X_{u,v}} = \underline{2} - \underline{1}$. If we assume in addition that $[K..L] \neq V$, then $K - 1 \neq L$ and we have $\underline{X(\mathcal{C})} = X_{K-1,K} + X_{L,L+1} = 2$, thus $\underline{\delta_u + \delta_v} - \underline{X_{u,v}} + \underline{X(\mathcal{C})} = 3 > 2$.
The set of points of $\widetilde{S}^n_{\delta,X}$ satisfying (H$_{\mathcal{P},1,n}$) to equality is thus exactly the following set.

$$\left\{ p^{K,L} \;\middle|\; (L, K) \in V^2 \setminus (W_1{}^{<} \sqcup W_2{}^{<}) \right\}$$

These points can be split into the five following types, depending on the value of parameters K and L regarding to u and v.

- $p^{K,L}$ for $(K, L) = (1, n)$
- $p^{K,L}$ for $(K, L) \in W_1{}^{\leqslant}$
- $p^{K,L}$ for $(K, L) \in W_2{}^{\leqslant}$
- $p^{K,L}$ for $(K, L) \in (W_2 \cup \{u\}) \times (\{u\} \cup W_1)$
- $p^{K,L}$ for $(K, L) \in (W_1 \cup \{v\}) \times (\{v\} \cup W_2)$

Using colors, Figure 4.6 gives an illustration of these five type of points.

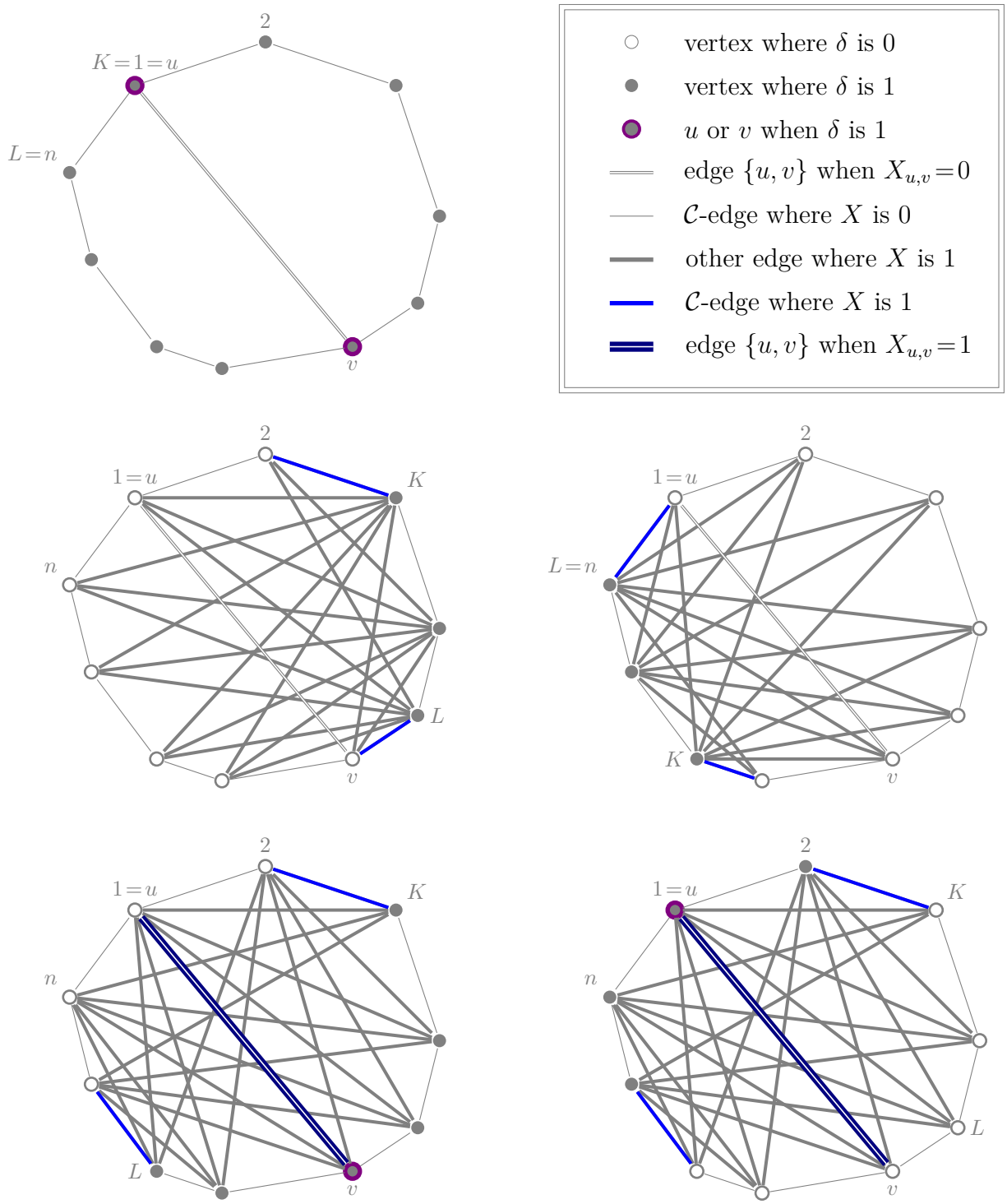


Figure 4.6: Illustration of the non-zero components of $p^{K,L}$, for the five different types of (K, L) regarding to u and v

For any $(L, K) \in V^2 \setminus (W_1{}^{<} \sqcup W_2{}^{<})$, $p^{K,L} \in F^a$ since $p^{K,L} \in F$, and we have $a.p^{K,L} = \alpha$, which can be written as follows.

$$a([K..L]) + a\Big([K..L] : (V \setminus [K..L])\Big) = \alpha \qquad (1_{K,L})$$

We now start to use equations $(1_{K,L})$ *for different values of* K *and* L*, which will allow to first prove that* $a_{/V} = \frac{\alpha}{2}(\mathbb{I}_u + \mathbb{I}_v)$.

– The equation $(1_{K,L})$ for $(K,L)=(1,n)$ gives:

$$a(W)+a_u+a_v = \alpha \tag{$1_{1,n}$}$$

– The equation $(1_{K,L})$ for $(K,L)=(1,1)=(u,u)$ gives:

$$a_u + a_{u,v} + a(\{u\}:W) = \alpha \tag{$1_{1,1}$}$$

while for $(K,L)=(2,n)$ it gives:

$$a(W) + a_v + a_{u,v} + a(\{u\}:W) = \alpha \tag{$1_{2,n}$}$$

These both equations imply that $a(W)+a_v=a_u$. Following the same line with v instead of u, we obtain $a(W)+a_u=a_v$. Then we have $a(W)=a_u-a_v=a_v-a_u$. We deduce that $a(W)=0$ and $a_u=a_v$. Using the equation $(1_{1,n})$, which is equivalent to $a_u+a_v=\alpha$, we finally have:

$$\boxed{a_u=a_v=\frac{\alpha}{2}}$$

– For any $J\in W_1$, the equation $(1_{K,L})$ for $(K,L)=(1,J)$ gives:

$$a([1..J]) + a([1..J]:[J+1..n]) = \alpha \tag{$1_{1,J}$}$$

while for $(K,L)=(J+1,n)$ it gives:

$$a([J+1..n]) + a([1..J]:[J+1..n]) = \alpha \tag{$1_{J+1,n}$}$$

These both equations imply that $a([1..J])=a([J+1..n])$. Besides, we have $u\in[1..J]$ and $v\in[J+1..n]$, since $J\in W_1$. Then we can substract $a_u=a_v$ on both sides, which is equivalent to only consider W nodes, to obtain the following equation.

$$a\Big([1..J]\cap W\Big)=a\Big([J+1..n]\cap W\Big) \tag{$\bigstar_J$}$$

Thanks to this latter equation, we will show that $a_J=0$ by induction on $J\in W_1=[2..v-1]$.

∘ *Initialization*
For $J=2$, the equation $(\bigstar_J)$ gives $a_2=a([3..n]\cap W)$, that is $a_2=a(W)-a_2$. Using that $a(W)=0$, we obtain $a_2=0$.
∘ *Induction*
Let $J\in[2..v-2]$. Let us assume that $\forall j\in[2..J]$, $a_j=0$.

On one hand, we then have:

$$a([1..J+1]\cap W) = \underbrace{a([2..J])}_{=0}+a_{J+1}$$

and on the other hand we also have:

$$a([J+2..n]\cap W) = a(W\setminus[2..J+1]) = \underbrace{a(W)}_{=0} - \underbrace{a([2..J+1])}_{=0} - a_{J+1}$$

The equation $(\bigstar_{J+1})$ is then equivalent to $a_{J+1}=-a_{J+1}$, that is $a_{J+1}=0$.
By induction, we have $a_J=0$ for any $J\in W_1$.
For any $J\in W_2$, we can similarly establish that:

$$a\Big([v..J]\cap W\Big)=a\Big([J+1..v-1]\cap W\Big)$$

and deduce by induction that $a_J=0$ for any $J\in W_2$. Finally, we have:

$$\boxed{\forall J\in W,\ a_J=0}$$

We continue using equations $(1_{K,L})$ *for different values of* K *and* L*, which will allow to prove that* $a_{/E} = \frac{\alpha}{2}(\mathbb{I}_{\mathcal{C}} - \mathbb{I}_{u,v})$ *i.e.* $\forall i \in [1..n[,\ a_{i,i+1} = \frac{\alpha}{2},\ a_{u,v} = -\frac{\alpha}{2}$ *and other* $a_e = 0$ *for* $e \in E \setminus \mathcal{C}$ *s.t.* $e \neq \{u, v\}$.

– For any $J \in V$. the equation $(1_{K,L})$ for $(K, L) = (J, J+1)$ gives:

$$a_J + a_{J+1} + \Big(a^-(J) + a^+(J) - a_{J,J+1}\Big) + \Big(a^-(J+1) - a_{J,J+1} + a^+(J+1)\Big) = \alpha \qquad (1_{J,J+1})$$

while for $(K, L) = (J, J)$ it gives:

$$a_J + \big(a^+(J) + a^-(J)\big) = \alpha \qquad (1_{J,J})$$

and for $(K, L) = (J+1, J+1)$ it gives:

$$a_{J+1} + \big(a^+(J+1) + a^-(J+1)\big) = \alpha \qquad (1_{J+1,J+1})$$

Substracting $(1_{J,J})$ and $(1_{J+1,J+1})$ to $(1_{J,J+1})$, we obtain $-2a_{J,J+1} = -\alpha$. We deduce that:

$$\boxed{\forall J \in V,\ a_{J,J+1} = \frac{\alpha}{2}}$$

– For any $J \in W_1 = [2..v-1]$, the equations $(1_{K,L})$ for $(K, L) = (1, J)$ and $(K, L) = (2, J)$ give:

$$a_u + a\big([2..J]\big) + a\big([1..J] : [J+1..n]\big) = \alpha \qquad (1_{1,J})$$

$$a\big([2..J]\big) + a\big([2..J] : [J+1..1]\big) = \alpha \qquad (1_{2,J})$$

Since $\forall w \in W,\ a_w = 0$, the equality of the both left members gives can be written as follows.

$$(a_u + 0) + a\left(\{1\} : [J+1..n]\right) + \cancel{a\left([2..J] : [J+1..n]\right)} = 0 + \cancel{a\left([2..J] : [J+1..n]\right)} + a\left([2..J] : \{1\}\right)$$

By adding $a\left(\{1\} : [2..J]\right)$ on the both sides, we obtain:

$$a_u + \underbrace{a\left(\{1\} : [J+1..n]\right) + a\left(\{1\} : [2..J]\right)}_{= a^+(1)} = 2\, a\left(\{1\} : [2..J]\right)$$

Since the equation $(1_{K,L})$ for $(K, L) = (1, 1) = (u, u)$ gives $a_u + a^+(1) = \alpha$, we deduce that:

$$\forall J \in [2..v-1],\ a\left(\{1\} : [2..J]\right) = \frac{\alpha}{2}$$

Since $a_{1,2} = \frac{\alpha}{2}$, that allow to show by a direct induction on J that:

$$\forall J \in [3..v-1],\ a_{u,J} = 0$$

Symmetrically, for any $J \in W_2$, using the equations $(1_{K,L})$ for $(K, L) = (J, n)$ and $(K, L) = (J, 1)$ we obtain:

$$\forall J \in [v+1..n-1],\ a_{u,J} = 0$$

Finally we have:

$$\boxed{\forall J \in W \setminus \{u-1, u+1\},\ a_{u,J} = 0}$$

Following the same line with v instead of u, we can show that:

$$\boxed{\forall J \in W \setminus \{v-1, v+1\},\ a_{v,J} = 0}$$

– The equation $(1_{K,L})$ for $(K,L)=(2,n)$, which is :

$$a([1..v-1])+a_v+a([v+1..n])+a(\{u\}:[2..v-1])+a_{u,v}+a(\{u\}:[v+1..n])=\alpha \qquad (1_{2,n})$$

can be written as follows using the previous result:

$$0+a_v+0+a_{1,2}+a_{u,v}+a_{n,1}=\alpha \qquad (1_{2,n})$$

Since we already have $a_v=a_{n,1}=a_{1,2}=\dfrac{\alpha}{2}$, we deduce:

$$\boxed{a_{u,v}=-\frac{\alpha}{2}}$$

– For any $J\in[1..v-2]$, the equations $(1_{K,L})$ for $(K,L)=(1,J)$ and $(K,L)=(1,J+1)$ give:

$$a([1..J])+a([1..J]:\{J+1\})+a([1..J]:[J+2..n])=\alpha \qquad (1_{1,J})$$

$$a([1..J])+a_{J+1}+a([1..J]:[J+2..n])+a(\{J+1\}:[J+2..n])=\alpha \qquad (1_{1,J+1})$$

Since $J+1\in W_1$, $a_{J+1}=0$, we deduce from these equations that:

$$a([1..J]:\{J+1\})=a(\{J+1\}:[J+2..n]) \quad \text{that is} \quad a^-(J+1)=a^+(J+1)$$

Moreover, we already have $a_{J+1}+a^+(J+1)+a^-(J+1)=\alpha$ $(1_{J+1,J+1})$ and $a_{J+1}=0$, since $J+1\in W_1\subseteq W$. We deduce that $a^-(J+1)=a^+(J+1)=\frac{\alpha}{2}$, and that for any $J+1\in[2..v]=W_1$.
Following the same line for $J+1\in[v+1..n-1]$ and using the equations $(1_{K,L})$ for $(K,L)=(v,J)$ $(K,L)=(v,J+1)$, we can show that $a^-(J+1)=a^+(J+1)=\frac{\alpha}{2}$, and that for any $J+1\in[v+1..n]=W_2$. Thus, we finally have:

$$\forall J\in W,\ a^-(J)=a^+(J)=\frac{\alpha}{2} \qquad (\clubsuit)$$

– Thanks to this latter equation, we will show that $\forall(I,J)\in {W_1}^2, J>I+1$, $a_{I,J}=0$, that is that the a component associated with an edge between two nodes in W_1 is zero, excepted if this edge belongs to $\mathcal{C}$. More precisely, assuming that $v\geqslant 5$ otherwise there is nothing to show, we will show by induction on $l\in[2..v-2]$ the following property:

$$H_l:\forall I\in W_1,\ \forall J\in[I+2..I+l]\cap W_1,\ a_{I,J}=0$$

∘ *Initialization*
Let start with $l=2$. Let $I\in W_1$ such that $I+l\in W_1$, *i.e.* $I\in[2..v-3]$. We show that $a_{I,I+2}=0$.
The equation $(1_{K,L})$ for $(K,L)=(I,I+2)$ gives:

$$a([I..I+2])+a([I..I+2]:[I+3..I-1])=\alpha \qquad (1_{I,I+2})$$

Since $[I..I+2]\subseteq W_1$, the first term is zero. The second one, which is then equal to α can be decomposed as follows.

$$\begin{aligned} a([I..I+2]:[I+3..I-1]) &= \left(a^-(I)+a^+(I)-a_{I,I+1}-a_{I,I+2}\right) \\ &+\left(a^-(I+1)-a_{I,I+1}+a^+(I+1)-a_{I,I+2}\right) \\ &+\left(a^-(I+2)-a_{I,I+2}-a_{I+1,I+2}+a^+(I+2)\right) \end{aligned}$$

Using $(\clubsuit)$ and that $a_{J,J+1}=\frac{\alpha}{2}$ for any $J\in V$, we obtain: $(\frac{\alpha}{2}-a_{I,I+2})+(0)+(\frac{\alpha}{2}-a_{I,I+2})=\alpha$, and deduce that $a_{I,I+2}=0$. Since this is true for any $I\in[2..v-3]$, the property H_2 stands.

∘ *Induction*

Let $l \in [2..v-3]$. Let us assume that the property H stands up to the rank l. Let $I \in [2..v-l-2]$, thus we have $I+l+1 \in W_1$. The equation $(1_{K,L})$ for $(K,L)=(I,I+l+1)$ gives:

$$a([I..I+l+1]) + a\left([I..I+l+1] : [I+l+2..I-1]\right) = \alpha \qquad (1_{I,I+l+1})$$

Since $[I..I+l] \subseteq W_1$, the first term is zero. The second one, which is then equal to α can be written $A+B$ where:

$$\begin{aligned}
A &= a\Big([I..I+l+1] : [I+l+2..n]\Big) \\
&= \sum_{k=I}^{I+l+1} a\Big(\{k\} : [I+l+2..n]\Big) \\
&= \left(\sum_{k=I}^{I+l-1} a^+(k) - a_{k,k+1} - a\Big(\{k\} : [k+2..I+l+1]\Big)\right) + \Big(a^+(I+l) - a_{I+l,I+l+1}\Big) + \ a^+(I+l+1) \\
&= \left(\sum_{k=I}^{I+l-1} \frac{\alpha}{2} - \frac{\alpha}{2} - \underbrace{a\Big(\{k\} : [k+2..I+l+1]\Big)}_{=0 \text{ except for } k=I}\right) + \Big(\frac{\alpha}{2} - \frac{\alpha}{2}\Big) + \frac{\alpha}{2} \text{ by } H_? \\
&= -\, a_{I,I+l+1} + \frac{\alpha}{2}
\end{aligned}$$

$$\begin{aligned}
B &= a\Big([I..I+l+1] : [1..I-1]\Big) \\
&= \sum_{k=I}^{I+l+1} a\Big(\{k\} : [1..I-1]\Big) \\
&= a^-(I) + \Big(a^-(I+1) - a_{I,I+1}\Big) + \left(\sum_{k=I+2}^{I+l+1} a^-(k) - a_{k-1,k} - a\Big(\{k\} : [I..k-2]\Big)\right) \\
&= \frac{\alpha}{2} + \Big(\frac{\alpha}{2} - \frac{\alpha}{2}\Big) + \left(\sum_{k=I+2}^{I+l+1} \frac{\alpha}{2} - \frac{\alpha}{2} - \underbrace{a\Big(\{k\} : [I..k-2]\Big)}_{=0 \text{ except for } k=I+l+1}\right) \text{ by } H_? \\
&= \frac{\alpha}{2} - a_{I,I+l+1}
\end{aligned}$$

The equation $(1_{I,I+l+1})$ is then equivalent to $(-a_{I,I+l+1} + \frac{\alpha}{2}) + (\frac{\alpha}{2} - a_{I,I+l+1}) = \alpha$ and thus $a_{I,I+l+1} = 0$. Since this is true for any $I \in [2..v-l-2]$ the property H_{l+1} stands.

Finally, we have shown by induction that:

$$\boxed{\forall (I,J) \in {W_1}^< \setminus \mathcal{C},\ \ a_{I,J} = 0}$$

Symmetrically, we can show that:

$$\boxed{\forall (I,J) \in {W_2}^< \setminus \mathcal{C},\ \ a_{I,J} = 0}$$

These two results can be summed up by $a_{I,J} = 0$ if I and J are nodes of W not consecutive on $\mathcal{C}$, assuming that they are both in W_1 or both in W_2. The rest of the proof is dedicated to show that $a_{u,v} = 0$ also for a pair of nodes in $W_1 \times W_2$.

– Let $I \in W_1$ and $J \in W_2$. The equations $(1_{K,L})$ for $(K,L)=(I,J)$ and $(K,L)=(I+1,J)$ can be written as follows.

$$\underline{a_I} + \underline{a([I+1..J])} + \underline{a(\{I\}:[J+1..I-1])} + \underline{a([I+1..J]:[J+1..I-1])} = \alpha \qquad (1_{I,J})$$

$$\underline{a([I+1..J])} + \underline{a([I+1..J]:[J+1..I-1])} + \underline{a([I+1..J]:\{I\})} = \alpha \qquad (1_{I+1,J})$$

Since $I \in W_1 \subseteq W_1$, we have $a_I = 0$. We then deduce from these both equation that:

$$a(\{I\}:[J+1..I-1]) = a([I+1..J]:\{I\})$$

Moreover, splitting the index interval between J and $J+1$ the equation $(1_{I,I})$ can be written as follows.

$$\underline{0} + \underline{a(\{I\}:[I+1..J])} + \underline{a(\{I\}:[J+1..I-1])} = \alpha \qquad (1_{I,I})$$

We deduce in particular that $a(\{I\}:[I+1..J]) = \frac{\alpha}{2}$. Since we already have:

- $a_{I,k} = 0$ for $k \in [I+2..v-1] \subseteq W^1$,
- $a_{I,v} = 0$ except if $v = I+1$,
- $a_{I,I+1} = \dfrac{\alpha}{2}$,

we first deduce that $a(\{I\}:[I+1..J]) = a_{I,I+1} + a(\{I\}:[v+1..J])$, and then that $a(\{I\}:[v+1..J]) = 0$. Using this latter equation iteratively for J going from $v+1$ to n, we deduce that $\forall w \in W_2, a_{I,w} = 0$. Since this is true for any $I \in W_1$, we deduce that:

$$\boxed{\forall (I,J) \in W_1 \times W_2,\ a_{I,J} = 0}$$

• *Conclusion*
Finally, $a = \dfrac{\alpha}{2}\left(\mathbb{I}_u + \mathbb{I}_v - \mathbb{I}_{u,v} + \sum_{i=1}^{n} \mathbb{I}_{(i,i+1)}\right)$. Since F^a defines a facets, $a \neq 0$, and thus $\alpha \neq 0$. Since dividing all the components of a by α does not change F^a, we can assume that $\alpha = 2$ without loss of generality. We then obtain $a = \mathbb{I}_u + \mathbb{I}_v - \mathbb{I}_{u,v} + \sum_{i=1}^{n} \mathbb{I}_{(i,i+1)}$. thus $F^a = F$ and F defines a facet. □

4.5 Without name inequalities

In this section, we assume that $n \geqslant 4$. For any node $(u,v) \in V^<$, we introduce the following inequality.

$$\delta_u + \delta_v - (n-3)X_{u,v} + X\Big(\{u,v\} : V \setminus \{u,v\}\Big) \geqslant 2 \qquad (?_{u,v})$$

Property 4.4

For any node $(u,v) \in V^<$, the inequality $(?_{u,v})$ defines a facet of $\widetilde{P}^n_{\delta,X}$

Proof: Let $(u,v) \in V^<$. Let us denote by W the set of the other nodes, *i.e.* $W = V \setminus \{u,v\}$. Inequality $(?_{u,v})$ can then be written as follows:

$$\underline{\delta_u} + \underline{\delta_v} - (n-3)\,\underline{X_{u,v}} + \underline{X(\{u\}:W)} + \underline{X(\{v\}:W)} \geqslant 2 \qquad (?_{u,v})$$

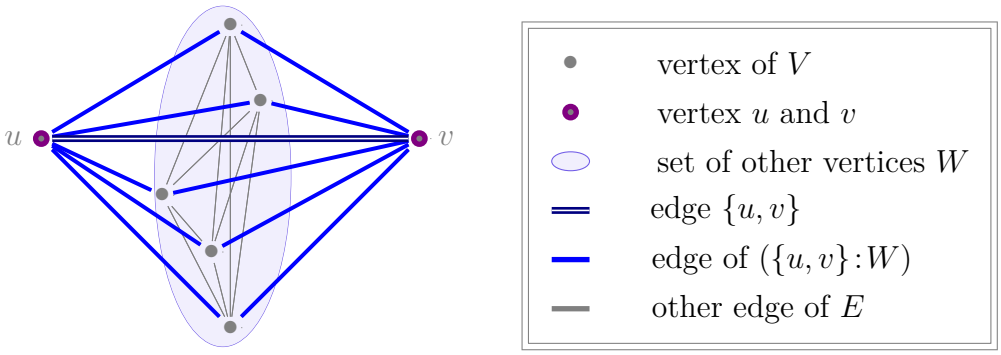


Figure 4.7: Illustration of the terms (δ, X) involved in $(?_{u,v})$

• *Validity*
Let $(\delta, X) \in \widetilde{S}^n_{\delta,X}$.

- If $(\delta_u, \delta_v) = (1,1)$ then $X_{u,v} = 0$ and $\delta_u + \delta_v - (n-3)\,X_{u,v} = 2$. Moreover, all X components are non-negative, therefore $\delta_u + \delta_v - (n-3)\,X_{u,v} + X(\{u,v\}:W) \geqslant 2$, that is (δ, X) satisfies inequality $(?_{u,v})$.

- If $(\delta_u, \delta_v) = (1,0)$ then $X_{u,v} = 1$, and $\delta_u + \delta_v - (n-3)X_{u,v} = 1-(n-3) = 4-n$.
Moreover, for any $i \in W$ we have $X_{i,u} = 1 - X_{i,v}$, thus $X(\{v\}:W) + X(\{v\}:W) = |W| = n-2$. We deduce that (δ, X) satisfies inequality $(?_{u,v})$ and even that it satisfies it to equality.

- If $(\delta_u, \delta_v) = (0,1)$, we show similarly that (δ, X) satisfies inequality $(?_{u,v})$ to equality.

- If $(\delta_u, \delta_v) = (0,0)$, then $X_{u,v} = 0$, and $\delta_u + \delta_v - (n-3)\,X_{u,v} = 0$. Moreover, since $\delta \neq 0$, there exists at least one node $i \in W$ such that $\delta_i = 1$, and thus such that $X_{i,u} = X_{i,v} = 1$. Therefore, we have $X(\{u\}:W) + X(\{v\}:W) \geqslant 2$. We deduce that (δ, X) satisfies inequality $(?_{u,v})$.

We deduce that all the points of $\widetilde{S}^n_{\delta,X}$ satisfy inequality $(?_{u,v})$. Since $\widetilde{P} = \mathrm{conv}(\widetilde{S}^n_{\delta,X})$, we deduce that any point in $\widetilde{P}^n_{\delta,X}$ satisfies also this inequality. In other words, $(?_{u,v})$ is valid for $\widetilde{P}^n_{\delta,X}$.

• *Introducing the including facet*
Let us consider $a \in \mathbb{R}^V \times \mathbb{R}^E$ and $\alpha \in \mathbb{R}$ such that $F^a = \{(\delta, X) \in \widetilde{P}^n_{\delta,X} \mid a \cdot (\delta, X) = \alpha\}$ is a facet of $\widetilde{P}^n_{\delta,X}$ containing the face F associated with $(?_{u,v})$, that is the face defined as follows.

$$F = \Big\{(\delta, X) \in \widetilde{P}^n_{\delta,X} \;\Big|\; \delta_u + \delta_v - (n-3)\,X_{u,v} + X(\{u\}:W) + X\,(\{v\}:W) = 2\Big\}$$

• *Integer points satisfying inequality* $(?_{u,v})$ *to equality*

For any $U \subseteq W$, let us introduce p_u^U (resp. p_v^U) the point of $\mathcal{S}$ defined by $\delta = \mathbb{I}_u + \mathbb{I}_U$ (resp. by $\delta = \mathbb{I}_v + \mathbb{I}_U$). As previously observed, such a point satisfies $(?_{u,v})$ to equality. Therefore, $p_u^U \in F \subseteq F^a$, (resp. $p_v^U \in F^a$) then we have $a.p_u^U = \alpha$ (resp. $a.p_u^U = \alpha$), which can be written as follows.

$$a_u + a(U) + a(\{u\}:W\setminus U) + a(\{v\}:U) + a(U:W\setminus U) + a_{u,v} = \alpha \qquad (1_U^u)$$

$$a_v + a(U) + a(\{u\}:U) + a(\{v\}:W\setminus U) + a(U:W\setminus U) + a_{u,v} = \alpha \qquad (1_U^v)$$

For any $J \in W$, let us introduce p^J the point of $\mathcal{S}$ defined by $\delta = \mathbb{I}_J$. Such a point also satisfies $(?_{u,v})$ to equality. Therefore, we have the following equation:

$$0 + a_J + a_{u,J} + a_{v,J} + a(\{J\}:W\setminus\{J\}) + 0 = \alpha \qquad (2_J)$$

Finally, the point of $\mathcal{S}$ defined by $\delta = \mathbb{I}_V$ also satisfies $(?_{u,v})$ to equality, and we deduce the following equation:

$$a_u + a_v + a(W) + 0 + 0 + 0 + 0 = \alpha \qquad (3)$$

Reasoning on the different cases for (δ_u, δ_v) as done to prove the validity of $(?_{u,v})$, we can prove that the set of points of $\mathcal{S}$ satisfying (Star_u) to equality is exactly $\left\{p_u^U, p_v^U \,\middle|\, U \subseteq W\right\} \sqcup \left\{p^J \,\middle|\, J \in W\right\} \sqcup \{(\mathbb{I}_V, 0)\}$. Figure 4.8 gives an illustration of the non-zero component of these four kinds of points.

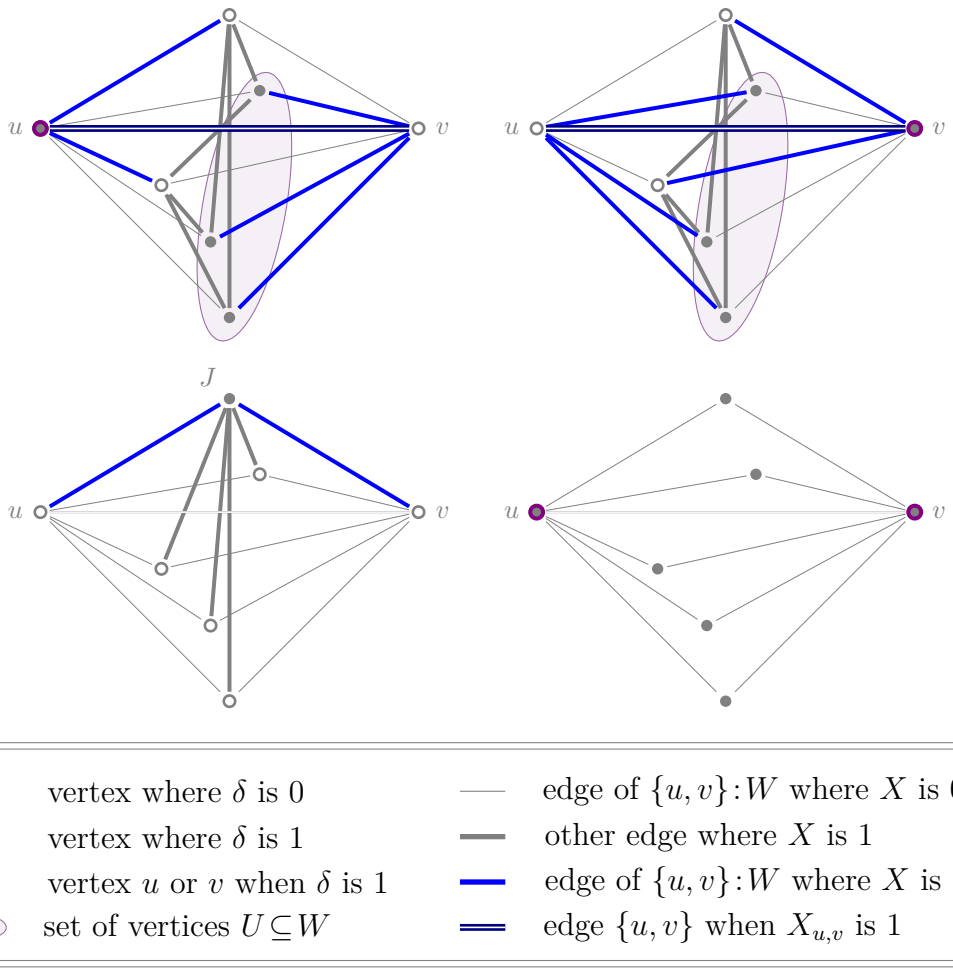


Figure 4.8: Illustration of the non-zero components of the integer points satisfying $(?_{u,v})$ to equality

We now start to use equations (1^u_U), (1^v_U), *and* (2_J) *for different values of* U *and* J, *which will allow to prove that* $a_{/V} = \frac{\alpha}{2}(\mathbb{I}_u + \mathbb{I}_v)$

– The equation (1^u_U) for $U = \emptyset$ gives:

$$a_u + 0 + a(\{u\}:W) + 0 + 0 + a_{u,v} = \alpha \qquad (1^u_\emptyset)$$

while the equation (1^u_U) for $U = \emptyset$ gives:

$$a_v + 0 + 0 + a(\{v\}:W) + 0 + a_{u,v} = \alpha \qquad (1^v_\emptyset)$$

We deduce that:

$$a_u + a(\{u\}:W) = a_v + a(\{v\}:W)$$

The equation (1^u_U) for $U = W$ gives:

$$a_u + a(W) + 0 + a(\{v\}:W) + 0 + a_{u,v} = \alpha \qquad (1^u_W)$$

while the equation (1^u_U) for $U = X$ gives:

$$a_v + a(W) + a(\{u\}:W) + 0 + 0 + a_{u,v} = \alpha \qquad (1^v_W)$$

We deduce that:

$$a_v + a(\{u\}:W) = a_u + a(\{v\}:W)$$

Thanks to these both equations, we obtain that $a_u = a_v$ and that $a(\{u\}:W) = a(\{v\}:W)$. Let us denote by S this latter quantity.

– For any $J \in W$, the equation (1^u_U) for $U = \{J\}$ can be written as follows:

$$a_u + a_J + \underbrace{a(\{u\}:W\setminus\{J\})}_{=S-a_{u,J}} + a_{v,J} + a(\{J\}:W\setminus\{J\}) + a_{u,v} = \alpha \qquad (1^u_{\{J\}})$$

while for $U = W\setminus\{J\}$ this equality can be written as follows:

$$a_u + a(W) - a_J + a_{u,J} + \underbrace{a(\{v\}:W\setminus\{J\})}_{=S-a_{v,J}} + a(\{J\}:W\setminus\{J\}) + a_{u,v} = \alpha \qquad (1^u_{W\setminus\{J\}})$$

Identifying the commmon terms in these both equations, we obtain:

$$a_J + \not{S} - a_{u,J} + a_{v,J} = a(W) - a_J + a_{u,J} + \not{S} - a_{v,J}$$

which is equivalent to $2(a_J - a_{u,J} + a_{v,J}) = a(W)$.

Following the same line with v instead of u, we obtain $2(a_J - a_{v,J} + a_{u,J}) = a(W)$.

Summing these both equation we deduce that $a_J = \frac{a(W)}{2}$, and substracting one to the other, we deduce that $a_{J,v} = a_{u,J}$. Since this is true for any $J \in W$, we have $a(W) = \sum_{J\in W} \frac{a(W)}{2} = |W| \frac{a(W)}{2}$. Since $|W| = n-2 \neq 0$, that implies that $a(W) = 0$, and then we have:

$$\boxed{\forall J \in W,\ a_J = 0}$$

Moreover, the equation (3) is then equivalent to $a_u + a_v = 0$. Since we already proved that $a_u = a_v$, we deduce that:

$$\boxed{a_u = a_v = \frac{\alpha}{2}}$$

We continue using equations (1^u_U), (1^v_U), *and* (2_J) *for different values of* U *and* J, *which will allow to prove that* $a_{/E} = \frac{\alpha}{2}(-(n-3)\mathbb{I}_{\{u,v\}} + \mathbb{I}_{\{u\}:W} + \mathbb{I}_{\{v\}:W})$

– For any $J \in W$, thanks to the previous results the equation (2_J) can be written as follows:

$$\underbrace{a_J}_{=0} + \underbrace{a_{u,J} + a_{v,J}}_{=2\,a_{u,J}} + a(\{J\}:W\setminus\{J\}) = \alpha \qquad (2_J)$$

while the equation $(1^u_{\{J\}})$ can be written as follows:

$$a_u + \underbrace{a_J}_{=0} + S \underbrace{- a_{u,J} + a_{v,J}}_{=0} + a(\{J\}:W\setminus\{J\}) + a_{u,v} = \alpha \qquad (1^u_{\{J\}})$$

We deduce from these both equations that $2\,a_{u,J} = a_u + a_{u,v} + S$. Since the equation (1^u_W) can also be simplified as follows,

$$a_u + \underbrace{a(W)}_{=0} + S + a_{u,v} = \alpha \qquad (1^u_W)$$

we obtain $2\,a_{u,J} = \alpha$, and finally we deduce that:

$$\boxed{\forall J \in W,\ a_{v,J} = a_{u,J} = \frac{\alpha}{2}}$$

Remembering that $S = a(\{u\}:W)$, we then have $S = |W|\,\frac{\alpha}{2} = (n-2)\,\frac{\alpha}{2}$ Since we already have $a_u = \frac{\alpha}{2}$, using once again the equation (1^u_W) allow to deduce that:

$$\boxed{a_{u,v} = -(n-3)\frac{\alpha}{2}}$$

– For any $(J,K) \in W^<$, using the previous results allows to simplify the equation (1^u_U) for $U = \{J,K\}$ as follows:

$$\underbrace{a_u}_{=\frac{\alpha}{2}} + \underbrace{a_J + a_K}_{=0} + \underbrace{a(\{u\}:W\setminus\{J,K\})}_{=(n-4)\frac{\alpha}{2}} + \underbrace{a_{v,J} + a_{v,K}}_{=\alpha} + a(\{J,K\}:W\setminus\{J,K\}) + \underbrace{a_{u,v}}_{=-(n-3)\frac{\alpha}{2}} = \alpha \quad (1^u_{\{J,K\}})$$

We deduce that $a(\{J,K\}:W\setminus\{J,K\}) = 0$.

Moreover, equation (1^u_U) for $U = \{J\}$ can also be simplified as follows:

$$\underbrace{a_u}_{=\frac{\alpha}{2}} + \underbrace{a_J}_{=0} + \underbrace{a(\{u\}:W\setminus\{J\})}_{=(n-3)\frac{\alpha}{2}} + \underbrace{a_{v,J}}_{=\frac{\alpha}{2}} + a(\{J\}:W\setminus\{J\}) + \underbrace{a_{u,v}}_{=-(n-3)\frac{\alpha}{2}} = \alpha \qquad (1^u_{\{J\}})$$

We deduce that $a(\{J\}:W\setminus\{J\}) = 0$, and similarly that $a(\{K\}:W\setminus\{K\}) = 0$.

Thanks to the following decomposition, that implies that $a_{J,K} = 0$.

$$\begin{aligned} a(\{J,K\}:W\setminus\{J,K\}) &= a(\{J\}:W\setminus\{J,K\}) + a(\{K\}:W\setminus\{J,K\}) \\ &= (a(\{J\}:W\setminus\{J\}) - a_{J,K}) + (a(\{K\}:W\setminus\{K\}) - a_{J,K}) \end{aligned}$$

Finally we have:

$$\boxed{\forall (J,K) \in W^<,\ a_{J,K} = 0}$$

• *Conclusion*

Finally, $a = \frac{\alpha}{2}(\mathbb{I}_u + \mathbb{I}_v - (n-3)\,\mathbb{I}_{u,v} + \mathbb{I}_{\{u\}:W} + \mathbb{I}_{\{v\}:W})$. Since F^a defines a facets, $a \neq 0$, and thus $\alpha \neq 0$. Since dividing all the components of a by α does not change F^a, we can assume that $\alpha = 2$ without loss of generality. We then obtain $a = \mathbb{I}_u + \mathbb{I}_v - (n-3)\,\mathbb{I}_{u,v} + \mathbb{I}_{\{u\}:W} + \mathbb{I}_{\{v\}:W}$ thus $F^a = F$ and F defines a facet. □

4.6 Star inequality

In this section, we assume that $n \geqslant 4$. For any node $u \in V$, we introduce the following inequality.

$$-(n-3)\,\delta_u + \delta\big(V\setminus\{u\}\big) + X\big(\{u\}:V\setminus\{u\}\big) \geqslant 2 \qquad (\text{Star}_u)$$

Property 4.5

For any node $u \in V$, the inequality (Star_u) defines a facet of $\widetilde{P}^n_{\delta,X}$

Proof: Let $u \in V$. Let us denote by W the set of the other nodes, *i.e.* $W = V\setminus\{u\}$. Inequality (Star_u) can then be written as follows:

$$-(n-3)\,\delta_u + \delta(W) + X(\{u\}:W) \geqslant 2 \qquad (\text{Star}_u)$$

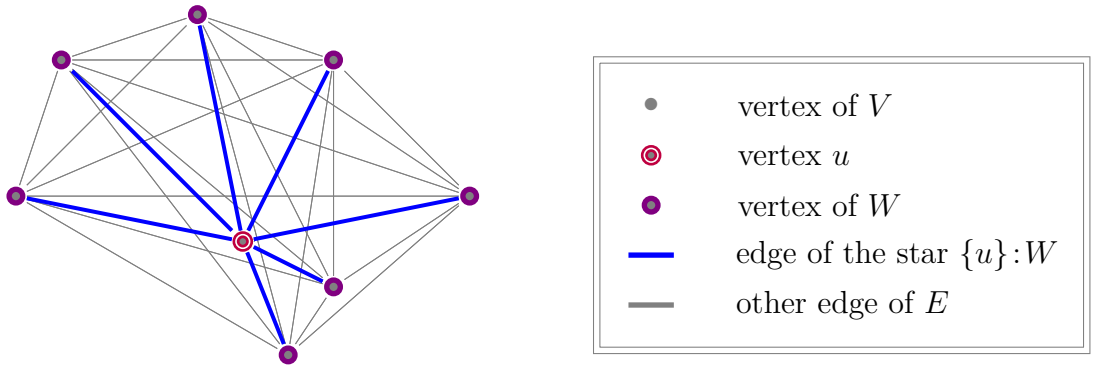


Figure 4.9: Illustration of the terms (δ, X) involved in (Star_u)

• *Validity*
Let $(\delta, X) \in \widetilde{S}^n_{\delta,X}$.

- If $\delta_u = 1$, then for every $i \in W$ we have $X_{\{i,u\}} = 1-\delta_i$, thus $\delta(W) + X(\{u\}:W) = |W| = n-1$. We deduce that (δ, X) satisfies (Star_u), and even that is satisfies it to equality.
- If $\delta_u = 0$, then there exists at least $i \in W$ such that $\delta_i = 1$ since $\delta \neq 0$. We then have $X_{\{i,u\}} = 1$, and since components of δ and X are non-negative, we deduce that (δ, X) satisfies (Star_u).

We deduce that all the points of $\widetilde{S}^n_{\delta,X}$ satisfy inequality (Star_u). Since $\widetilde{P}^n_{\delta,X} = \text{conv}(\widetilde{S}^n_{\delta,X})$, we deduce that any point in $\widetilde{P}^n_{\delta,X}$ satisfies also this inequality. In other words, (Star_u) is valid for $\widetilde{P}^n_{\delta,X}$.

• *Introducing the including facet*
Let us consider $a \in \mathbb{R}^V \times \mathbb{R}^E$ and $\alpha \in \mathbb{R}$ such that $F^a = \{(\delta, X) \in \widetilde{P}^n_{\delta,X} \mid a\cdot(\delta, X) = \alpha\}$ is a facet of $\widetilde{P}^n_{\delta,X}$ containing the face F associated with (Star_u), that is the face defined as follows.

$$F = \{(\delta, X) \in \widetilde{P}^n_{\delta,X} \mid -(n-3)\delta_u + \delta(W) + X(\{u\}:W) = 2\}$$

• *Integer points satisfying inequality (Star_u) to equality*
For any $U \subseteq W$, let us denote by p^U the point of $\mathcal{S}$ defined by $\delta = \mathbb{I}_u + \mathbb{I}_U$. As previously said, such a point satisfies (Star_u) to equality, since $\delta_u = 1$. Therefore, $p^U \in F \subseteq F^a$, then we have $a.p^U = \alpha$, which can be written as follows.

$$a_u + a(U) + a(\{u\}:W\setminus U) + a(U:W\setminus U) = \alpha \qquad (1_U)$$

Moreover, for any $J \in W$, let us denote by p^J the point of $\mathcal{S}$ defined by $\delta = \mathbb{I}_J$. Such a point satisfies also (Star_u) to equality, since x has then exactly component equal to 1: $X_{u,J}$. Therefore, $p^U \in F \subseteq F^a$, then we have $a.p^U = \alpha$, which can be written as follows.

$$a_J + a_{u,J} + a(\{J\}:W\setminus\{J\}) = \alpha \tag{2_J}$$

Note that the set of points of $\mathcal{S}$ satisfying (Star_u) to equality is exactly $\left\{p^U \,\middle|\, U \subseteq W\right\} \sqcup \left\{p^J \,\middle|\, J \in W\right\}$. Figure 4.10 gives an illustration of the non-zero components of these two kinds of points.

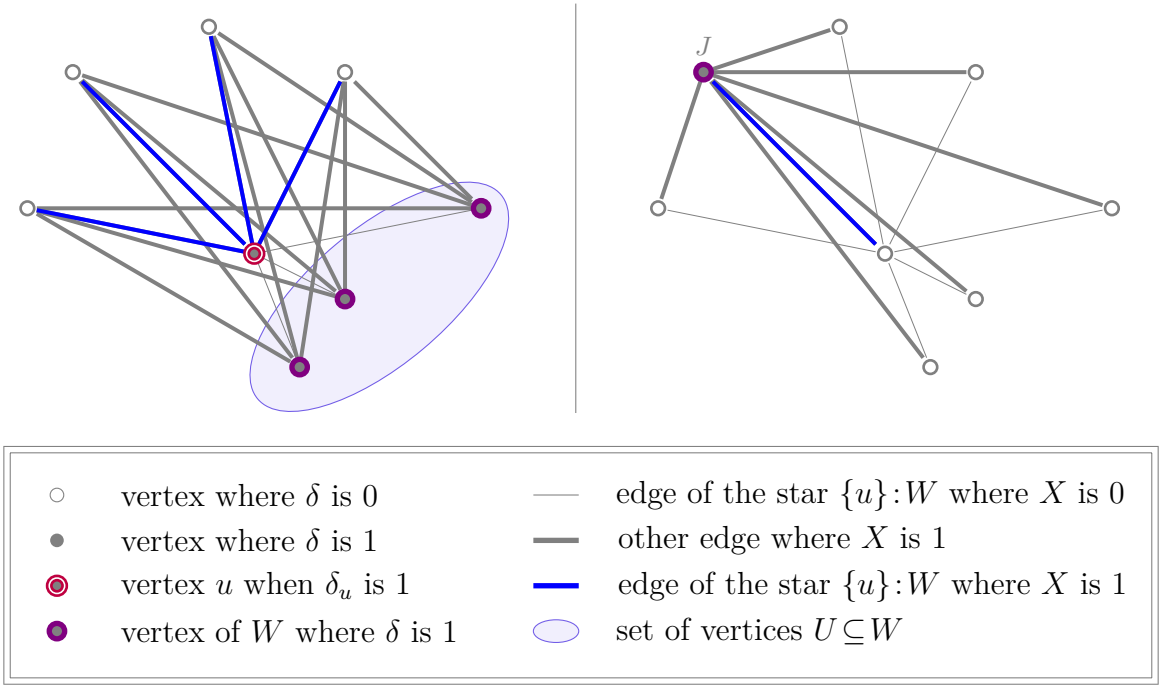


Figure 4.10: Illustration of the non-zero component of p^U (left) and p^J (right)

We now start to use equations (1_U) *and* (2_J) *for different values of* U *and* J*, which will allow to first prove that* $a_{/V} = \frac{\alpha}{2}((n-3)\mathbb{I}_u + \mathbb{I}_W)$.

– The equation (1_U) for $U = W$ gives:

$$a_u + a(W) + 0 + 0 = \alpha \tag{1_W}$$

The equation (1_U) for $U = \emptyset$ gives:

$$a_u + 0 + a(\{u\}:W) + 0 = \alpha \tag{$1_\emptyset$}$$

We deduce from these both equations that $a(W) = a(\{u\}:W)$ Let us denote by S this quantity. We then have $a_u + \mathcal{S} = \alpha$.

– For any $J \in W$, the equation (1_U) for $U = W\setminus\{J\}$ gives:

$$a_u + (S - a_J) + a_{u,J} + a(\{J\}:W\setminus\{J\}) = \alpha \tag{$1_{W\setminus\{J\}}$}$$

By substracting (2_J) to this expression we obtain $a_u + (S - 2\,a_J) = 0$, which is equivalent to $\alpha = 2\,a_J$ since $\alpha = a_u + S$. We finally deduce that:

$$\boxed{\forall J \in W,\ a_J = \frac{\alpha}{2}}$$

– The quantity S is then equal to $a(W) = \frac{\alpha}{2}|W| = (n-1)\frac{\alpha}{2}$. Using that $a_u = 2\frac{\alpha}{2} - S$ we deduce that:

$$\boxed{a_u = -(n-3)\frac{\alpha}{2}}$$

We continue using equations (1_U) and (2_J) for different values of U and J, which will allow to prove that $a_{/E} = \frac{\alpha}{2}\mathbb{I}_{\{u\}:W}$

– For any $J \in W$, the equation (1_U) for $U = \{J\}$ can be written as follows:

$$a_u + a_J + (S - a_{u,J}) + a(\{J\}:W\setminus\{J\}) = \alpha \qquad (1_{\{J\}})$$

By substracting (2_J) to this expression we obtain $a_u + \left(S - 2\,a_{u,J}\right) = 0$, which is equivalent to $\alpha = 2\,a_{u,J}$ since $\alpha = a_u + S$. We finally deduce that:

$$\boxed{\forall J \in W,\ a_{u,J} = \frac{\alpha}{2}}$$

– For any $J \in W$, the equation (2_J) can then be simplified as follows:

$$\frac{\alpha}{2} + \frac{\alpha}{2} + a(\{J\}:W\setminus\{J\}) = \alpha \qquad (2_J)$$

We deduce that: $\forall J \in W,\ a(\{J\}:W\setminus\{J\}) = 0$.

– For any $(J,K) \in W^<$, the equation (1_U) for $U = W\setminus\{J,K\}$ can be written as follows:

$$a_u + (S - a_J - a_K) + (a_{u,J} + a(\{J\}:W\setminus\{J\}) - a_{J,K}) + (a_{u,K} + a(\{K\}:W\setminus\{K\}) - a_{J,K}) = \alpha \quad (1_{W\setminus\{J,K\}})$$

Using the previous results, this equation can be simplified as follows:

$$\underbrace{a_u + (S}_{=\alpha} - \frac{\alpha}{2} - \frac{\alpha}{2}) + (\frac{\alpha}{2} + 0 - a_{J,K}) + (\frac{\alpha}{2} + 0 - a_{J,K}) = \alpha \qquad (1_{W\setminus\{J,K\}})$$

We deduce that

$$\boxed{\forall (J,K) \in W^<,\ a_{J,K} = 0}$$

• *Conclusion*
Finally, $a = \frac{\alpha}{2}\left(-(n-3)\mathbb{I}_u + \mathbb{I}_W + \mathbb{I}_{\{u\}:W}\right)$. Since F^a defines a facets, $a \neq 0$, and thus $\alpha \neq 0$. Since dividing all the components of a by α does not change F^a, we can assume that $\alpha = 2$ without loss of generality. We then obtain $a = -(n-3)\mathbb{I}_u + \mathbb{I}_W + \mathbb{I}_{\{u\}:W}$. thus $F^a = F$ and F defines a facet. □

4.7 Full inequalities

Let us assume for this section that $n \geqslant 5$ For any $(u,v) \in V^<$, let us introduce the following inequality:

$$(n\text{–}3)\Big(\delta\big(V_{\setminus\{u,v\}}\big) + X\big(\{u,v\} : V_{\setminus\{u,v\}}\big)\Big) - \Big(\frac{1}{2}n^2 - \frac{7}{2}n + 6\Big)\big(\delta_u + \delta_v + X_{u,v}\big) - X\big(V^<_{\setminus\{u,v\}}\big) \geqslant 2n\text{–}6 \quad (\text{Full}_{u,v})$$

Property 4.6

For any $(u,v) \in V^<$, the inequality $(\text{Full}_{u,v})$ is valid for $\widetilde{P}^n_{\delta,X}$.
Moreover, this inequality defines a facet of $\widetilde{P}^n_{\delta,X}$ if and only if $n = 5$.

Proof: Let $(u,v) \in V^<$ and $W = V \setminus \{u,v\}$. We then have $|W| = n-2$, and hence, $(|W|-1)\,|W| = n^2 - 5n + 6$. Let $Q = \frac{(|W|-3)|W|}{2} + 1 = 0.5n^2 - 3.5n + 6$. We then have $(|W|-1)\,|W| - 2Q = 2n-6$ and inequality $(\text{Full}_{u,v})$ can be written as follows.

$$(|W|-1)\Big(\delta(W) + X(\{u\}:W) + X(\{v\}:W)\Big) - Q\big(\delta_u + \delta_v + X_{u,v}\big) - X(W^<) \geqslant (|W|-1)\,|W| - 2Q \quad (\text{Full}_{u,v})$$

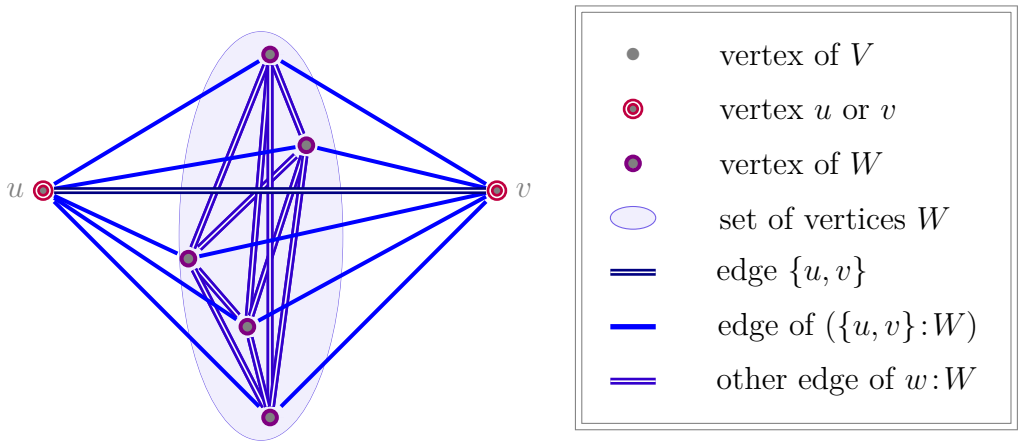


Figure 4.11: Illustration of the terms (δ, X) involved in $(\text{Full}_{u,v})$

• *Validity*
Let us prove that this inequality is valid for $\widetilde{P}^n_{\delta,X}$. Let $(\delta, X) \in \widetilde{S}^n_{\delta,X}$. Note that, by definition of $\widetilde{S}^n_{\delta,X}$, $X(W^<)$ is the number of edges in $W^<$ which links two nodes having a different δ value. Since the considered graph is complete, $X(W^<) = \delta(W)\,(1-\delta)(W)$.

- If $(\delta_u, \delta_v) = (1,1)$ then $X_{u,v} = 0$, thus we have $(\delta_u + \delta_v + X_{u,v}) = 2$.
  Moreover, for any $w \in W$, $X_{u,w} = X_{v,w} = 1 - \delta_w$, thus we have $X(\{u\}:W) = X(\{v\}:W) = (1-\delta)(W)$.
  Since $\delta(W) + (1-\delta)(W) = |W|$ inequality $(\text{Full}_{u,v})$ can be simplified as follows.

$$\begin{aligned}
(\text{Full}_{u,v}) &\Leftrightarrow (|W|-1)\Big(|W| + (1-\delta)(W)\Big) - \cancel{2Q} - X(W^<) \geqslant (|W|-1)\,|W| - \cancel{2Q} \\
&\Leftrightarrow (|W|-1)\Big(\cancel{|W|} + (1-\delta)(W)\Big) - X(W^<) \geqslant \cancel{(|W|-1)\,|W|} \\
&\Leftrightarrow X(W^<) \leqslant (|W|-1)(1-\delta)(W) \\
&\Leftrightarrow \delta(W)\,(1-\delta)(W) \leqslant (|W|-1)(1-\delta)(W)
\end{aligned}$$

This latter inequality is true no matter what is $\delta_{/W}$, since $a(N-a) \leqslant (N-1)(N-a)$ is true for any $N \in \mathbb{N}^*$ and $a \in [1..N]$, and hence in particular for $N = |W|$ and $a = \delta(W)$. We deduce that (δ, X) satisfies $(\text{Full}_{u,v})$.

- If $(\delta_u, \delta_v) = (1,0)$ then $X_{u,v} = 1$, thus we have $(\delta_u + \delta_v + X_{u,v}) = 2$.
  Moreover, for any $w \in W$, $X_{u,w} 1 - \delta_w =$ while $X_{v,w} = \delta_w$, thus we have $X(\{u\}:W) = (1-\delta)(W)$ and $X(\{v\}:W) = \delta(W)$. Inequality $(\text{Full}_{u,v})$ can be simplified as follows.

$$\begin{aligned}(\text{Full}_{u,v}) &\Leftrightarrow (|W|-1)(|W|+\delta(W)) - 2Q - X(W^<) \geqslant (|W|-1)\,|W| - 2Q \\ &\Leftrightarrow (|W|-1)(|W|+\delta(W)) - X(W^<) \geqslant (|W|-1)\,|W| \\ &\Leftrightarrow X(W^<) \leqslant (|W|-1)\delta(W) \\ &\Leftrightarrow \delta(W)\,(1-\delta)(W) \leqslant (|W|-1)\delta(W)\end{aligned}$$

  By the same arguments as previously, (with $a = (1-\delta)(W)$ this time), this latter inequality is true no matter what is $\delta_{/W}$. We deduce that (δ, X) satisfies $(\text{Full}_{u,v})$.

- If $(\delta_u, \delta_v) = (0,1)$, proving that (δ, X) satisfies $(\text{Full}_{u,v})$ follows the same line.

- If $(\delta_u, \delta_v) = (0,0)$, then $X_{u,v} = 0$, thus we have $(\delta_u + \delta_v + X_{u,v}) = 0$.
  Moreover, for any $w \in W$, $X_{u,w} = X_{v,w} = \delta_w$, thus we have $X(\{u\}:W) = X(\{v\}:W) = \delta(W)$. Remembering that $|W| = n$–2 and $(|W|$–$1)\,|W| - 2Q = 2n$–6, inequality $(\text{Full}_{u,v})$ can be transformed as follows.

$$\begin{aligned}(\text{Full}_{u,v}) &\Leftrightarrow (|W|-1)\,(3\,\delta(W)) - X(W^<) - 0Q \geqslant 2n-6 \\ &\Leftrightarrow 3\,\delta(W)(|W|-1) - (\delta(W)\,(1-\delta)(W)) \geqslant 2|W| - 2\end{aligned}$$

  This latter inequality is true no matter what is $\delta_{/W}$, excepted 0.
  Indeed, the function $f = a \mapsto 3(N-1)a - a(N-a)$ is strictly increasing for any $N \in \mathbb{N}^*$. and hence in particular for $N = |W| \geqslant 3$. We then have $f(a) \geqslant f(1) = 2N-2$ for any $a \in [1..N]$, and hence in particular for $a = \delta(W)$ which is not zero since $\delta \neq 0$ and $\delta_{/\overline{W}} = (\delta_u, \delta_v) = 0$.
  We deduce (δ, X) satisfies $(\text{Full}_{u,v})$.

These four items show that any point in $\widetilde{S}^n_{\delta,X}$ satisfies $(\text{Full}_{u,v})$. Since $\widetilde{P}^n_{\delta,X} = \text{conv}(\widetilde{S}^n_{\delta,X})$, we deduce that any point in $\widetilde{P}^n_{\delta,X}$ satisfies also this inequality. In other words, $(\text{Full}_{u,v})$ is valid for $\widetilde{P}^n_{\delta,X}$.

• *Integer points satisfying* $(\text{Full}_{u,v})$ *to equality*
One can check that the points $(\delta, X) \in \widetilde{P}^n_{\delta,X}$ satisfying $(\text{Full}_{u,v})$ to equality are exactly the $3\,(n$–$1)+(n$–$2) = 4n-5$ points given by the following δ.

- $\delta = \mathbb{I}_u + \mathbb{I}_v + \mathbb{I}_W$
- $\delta = \mathbb{I}_u + \mathbb{I}_v + \mathbb{I}_{W\setminus\{w\}}$ for any $w \in W$
- $\delta = \mathbb{I}_u + \mathbb{I}_w$ for any $w \in W$
- $\delta = \mathbb{I}_u$
- $\delta = \mathbb{I}_v + \mathbb{I}_w$ for any $w \in W$
- $\delta = \mathbb{I}_v$
- $\delta = \mathbb{I}_w$ for any $w \in W$

This is true for any $n \geqslant 5$, but as soon as $n > 5$, $4n-5 < n\frac{n+1}{2}$. We deduce that inequality $(\text{Full}_{u,v})$ does not define a facet for $n > 5$. Indeed, the dimension of a face is the dimension of the space generated by its extreme points, which are also extreme points for the polyhedron. The extreme points of the face defined by $(\text{Full}_{u,v})$ are then the above listed points, and they are too few to generate a $n\frac{n+1}{2}$ -dimensional space.

From now on, let us fix $n=5$.

• *Introducing the including facet*
Let us consider $a\in\mathbb{R}^V\times\mathbb{R}^E$ and $\alpha\in\mathbb{R}$ such that $F^a=\{(\delta,X)\in\widetilde{P}^n_{\delta,X}\,|\,a\cdot(\delta,X)=\alpha\}$ is a facet of $\widetilde{P}^n_{\delta,X}$ containing the face F associated with (Star_u), that is the face defined as follows.

$$F=\Big\{(\delta,X)\in\widetilde{P}^n_{\delta,X}\;\Big|\;(|W|-1)\Big(\delta(W)+X(\{u,v\}:W)\Big)-Q\Big(\delta_u+\delta_v+X_{u,v}\Big)-X(W^<)=(|W|-1)\,|W|-2Q\Big\}$$

– Let $w\in W$. By considering $(\delta,X)\in\mathcal{S}^n$ such that $\delta=\mathbb{I}_u+\mathbb{I}_v+\mathbb{I}_{W\setminus w}$. we obtain:

$$a_u+a_v+a(W)-a_w+a_{u,w}+a_{v,w}+a\big(\{w\}:W\setminus\{w\}\big)=\alpha \tag{4.1}$$

while by considering $(\delta,X)\in\mathcal{S}^n$ such that $\delta=\mathbb{I}_u+\mathbb{I}_v+\mathbb{I}_W$, we obtain:

$$a_u+a_v+a(W)=\alpha \tag{4.2}$$

Doing the difference, we deduce that:

$$a\big(\{w\}:W\setminus\{w\}\big)=a_w-a_{u,w}-a_{v,w} \tag{4.3}$$

By considering $(\delta,X)\in\mathcal{S}^n$ such that $\delta=\mathbb{I}_w$ we obtain:

$$a_w+a_{u,w}+a_{v,w}+a\big(\{w\}:W\setminus\{w\}\big)=\alpha \tag{4.4}$$

Remplacing $a\big(\{w\}:W\setminus\{w\}\big)$ according to (4.3), we deduce that $a_w=\frac{\alpha}{2}$.

$$\boxed{\forall w\in W,\ a_w=\frac{\alpha}{2}}$$

By summing up on W, we then have $a(W)=|W|\,\dfrac{\alpha}{2}=\dfrac{3\alpha}{2}$ and (4.2) can be written as follows.

$$a_u+a_v=-\frac{\alpha}{2} \tag{4.5}$$

– Let $w\in W$. By considering the point $(\delta,X)\in\mathcal{S}^n$ such that $\delta=\mathbb{I}_u+\mathbb{I}_w$, we obtain:

$$a_u+a_w+a_{u,v}+a_{v,w}+a\big(\{u\}:W\big)-a_{u,w}+a\big(\{w\}:W\setminus\{w\}\big)=\alpha \tag{4.6}$$

while by considering the point $(\delta,X)\in\mathcal{S}^n$ such that $\delta=\mathbb{I}_u+\mathbb{I}_v+\mathbb{I}_W$, we obtain:

$$a_u+a_{u,v}+a\big(\{u\}:W\big)=\alpha \tag{4.7}$$

Doing the difference, we deduce that:

$$a\big(\{w\}:W\setminus\{w\}\big)=-a_w-a_{v,w}+a_{u,w} \tag{4.8}$$

By gathering (4.3) and (4.8), we obtain $a_{u,w}=a_w=\frac{\alpha}{2}$.

$$\boxed{\forall w\in W,\ a_{u,w}=\frac{\alpha}{2}}$$

By summing up on W, we then have $a\big(\{u\}:W\big)=\dfrac{3\alpha}{2}$, and (4.7) can be written as follows.

$$a_u+a_{u,v}=-\frac{\alpha}{2} \tag{4.9}$$

– Following the same line with v instead of u, we obtain:

$$\boxed{\forall w \in W,\ a_{v,w} = \frac{\alpha}{2}} \text{ and } a_v + a_{u,v} = -\frac{\alpha}{2}$$

That implies in particular that $a_u = a_v$. We deduce from (4.5) that:

$$\boxed{a_u = a_v = -\frac{\alpha}{4}}$$

and from (4.9) that:

$$\boxed{a_{u,v} = -\frac{\alpha}{4}}$$

Morever, that allows also to simplify (4.3) for any $w \in W$:

$$a(\{w\} : W \setminus \{w\}) = -\frac{\alpha}{2}$$

This equation ensures that the weight of two arbitrary edges among the three edges in $W^<$ is always the same. That implies that the weight of the third edge is always the same. Therefore, the weight of each edge is the half of the weight for two edges.

$$\boxed{\forall (w, w') \in W^<,\ a_{w,w'} = -\frac{\alpha}{4}}$$

• *Conclusion*
Finally, $a = \frac{\alpha}{2}\Big(\mathbb{I}_W - \frac{1}{2}(\mathbb{I}_u + \mathbb{I}_v) + \mathbb{I}_{\{u,v\}:W} - \frac{1}{2}(\mathbb{I}_{W^<} + \mathbb{I}_{u,v})\Big)$. Since F^a defines a facet, $a \neq 0$, and thus $\alpha \neq 0$. Since dividing all the components of a by α does not change F^a, we assume that $\alpha = 4$ without loss of generality. We then obtain $a = 2\,\mathbb{I}_W - (\mathbb{I}_u + \mathbb{I}_v) + 2\,\mathbb{I}_{\{u,v\}:W} - (\mathbb{I}_{W^<} + \mathbb{I}_{u,v})\Big)$, thus $F^a = F$ and F defines a facet. □

The large number of facets appearing in $\widetilde{P}^n_{\delta,X}$ and also the variety of the facet defining inequalities of $\widetilde{P}^n_{\delta,X}$ dissuade to use them to reinforce F^2_l in practice. Indeed, to be efficient, such inequality families should be managed by a separation algorithm, and probably a dedicated algorithm for each inequalities family, since they do not seems to share a common structure.
Therefore, we investigate in the following part some inequalities able to cut off integer points having a high value according to the objective function, without necessarily being facet defining. A priori, such inequalities would maybe have a lower impact on the linear relaxation value, since they are not tight, but would be more efficient in practice.

PART C

Dominance inequalities

In this part, we propose linear inequalities to reinforce MIP formulations. These inequalities depend on the feasible solution set, but also on the objective function, since they cut off integer points encoding locally non-optimal solutions. Therefore, they differ from the classical strengthening inequalities, which cut off fractional points, and from symmetry breaking inequalities, which cut off integer points which may be optimal. In contrast with Chapter 4, we do not focus on facet-defining inequalities in this part.

We call **dominance inequalities** such inequalities, since they translate the dominance of locally optimal solutions for any given neighborhood. While local search methods eliminate locally non-optimal solutions when exploring the solution space, the dominance inequalities operate *globally* and *a priori*. Actually, each inequality is associated with a well chosen operation on the solutions and cuts off, at one stroke, all the solutions that can be improved by applying this operation.

Part C is divided into three chapters. In Chapter 5, we propose two families of dominance inequalities for Formulation F^2. They are based on the so-called insert and swap operations, which can be viewed as operations on the partitions or as operations on the schedules. In Chapter 6, we explain how these inequalities (resp. these operations) can be used for solving exactly (resp. heuristically) UCDDP. Experimental results are provided. In Chapter 7, we present ongoing works on dominance inequalities for other combinatorial optimization problems.

Chapter 5

Dominance inequalities for UCDDP

In Section 5.1, we present a generic dominance property to be instantiated by a neighborhood function. We focus a neighborhood function based on operations, and present two kinds of operations on the ordered bi-partitions: the insert and swap operations. In Section 5.2, we explain how to obtain a linear inequality translating the dominance property associated with the insert operations. In Section 5.3, we attempt to generalize this approach, and apply the proposed generic framework to swap operations in Section 5.4. Finally, in Section 5.5, we provide some results about the insert and swap inequalities.

5.1 Neighborhood based dominance properties

• *Neighborhood based dominance properties*

Let us recall some definitions used in local search context [3]. A **neighborhood function** $\mathcal{N}$ is a function which associates to any solution S a subset of solutions $\mathcal{N}(S)$, called the **neighborhood** of S. A solution of $\mathcal{N}(S)$ is a **neighbor** of S. Moreover, a solution S is **locally optimal** with respect to minimizing function f^{obj} if $f^{obj}(S) \leqslant f^{obj}(S')$, for any neighbor $S' \in \mathcal{N}(S)$, otherwise, S is **dominated** (by S'). If needed, we will say that S is **$\mathcal{N}$-locally optimal** or **$\mathcal{N}$-dominated**.

Given a neighborhood function, the set of locally optimal solutions always contains all optimal solutions, and is therefore a strictly dominant set. This statement can be seen as a generic dominance property. This kind of dominance property is commonly used in local search. Indeed, a step of a local search procedure consists in enumerating some of the neighbors of a given solution S, computing their values and then, if a better solution is found, moving to the best solution found at the current iteration, which is equivalent to discard S as it is dominated.

Definition 5.1

Let $\mathcal{I}$ be a family of linear inequalities of $\mathbb{R}^{\dim}$. Let $\mathscr{S}' \subseteq \mathscr{S}$ be a dominant set.

We say that $\mathcal{I}$ ***translates the dominance of*** *$\mathscr{S}'$ if* $\begin{cases} \forall S \in \mathscr{S} \setminus \mathscr{S}', \exists I \in \mathcal{I}, \text{ s.t. } S \text{ does not satisfy } I \\ \forall S \in \mathscr{S}', \forall I \in \mathcal{I}, S \text{ satisfies } I \end{cases}$

Such kind of inequalities will be called ***dominance inequalities****.*
If $\mathscr{S}'$ is the set of the $\mathcal{N}$-local optima, we say that $\mathcal{I}$ translates the $\mathcal{N}$-dominance property.

• *Operation based neighborhood functions*

We call **operation** any (possibly partial) function, which maps a solution to another solution. In this work, we will consider neighborhood functions based on a set of operations. The neighborhood of a solution S is then the set of the solutions obtained by applying to S any operation defined on S.

This kind of neighborhood functions allows to use the generic dominance property in a different way. Instead of considering sequentially the neighborhood of each solution, we will consider sequentially each operation. For each one, each solution is compared to its neighbor obtained using this operation. This differentiates what we call the solution-centered and the operation-centered point of view (see Section 5.3.1).

• *Insert and swap operations*

In the case of problems where the solutions are partitions, two operations are commonly considered to define a neighborhood: the **insertion**, which consists in moving an element from a subset to the other, and the **swap** which consists in swapping two elements of two different subsets. Figure 5.1 illustrates these operations for an ordered bi-partition (E, T).

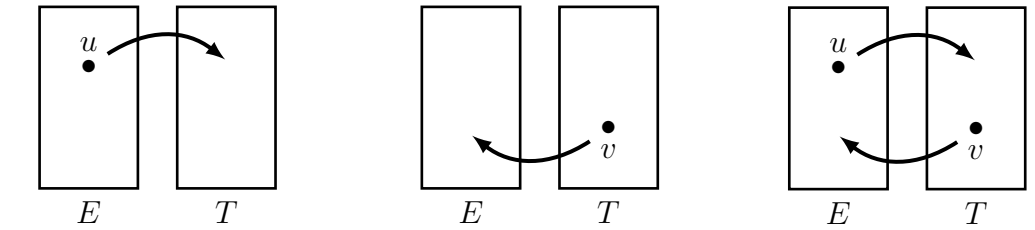


Figure 5.1: Illustration of swap and insert operations on the ordered bi-partitions

In the context of common due date scheduling, the insert operation consists in removing a task $j \in J$ from the early (resp. tardy) side and inserting j on the tardy (resp. early) side, as early (resp. as tardy) as possible according to its β-ratio (resp. α-ratio). The tasks scheduled before (resp. after) j are right-shifted (resp. left-shifted) by p_j time units. The swap operation consists in sequentially inserting an early task on the tardy side and inserting another tardy task on the early side, or vice-versa, as described above. Let us denote by $\text{insert}_u(\mathcal{S})$ (resp. $\text{swap}_{u,v}(\mathcal{S})$) the schedule obtained from a schedule $\mathcal{S}$ by the insert (resp. swap) operation on task u (resp. on tasks u, v).

Since the insert and swap operations are fundamentally defined on partitions (*i.e.* over equivalence classes of V-shaped d-blocks), these operations preserve the equivalence. Therefore, we have $\forall u \in J$, $\mathcal{S} \sim \mathcal{S}' \Rightarrow \text{insert}_u(\mathcal{S}) \sim \text{insert}_u(\mathcal{S}')$ and $\forall (u, v) \in J^2$, $\mathcal{S} \sim \mathcal{S}' \Rightarrow \text{swap}_{u,v}(\mathcal{S}) \sim \text{swap}_{u,v}(\mathcal{S}')$, assuming that u and v are not on the same side in the schedules $\mathcal{S}$ and $\mathcal{S}'$.

Let us define local optima for the neighborhoods related to these operations in order to enunciate the associated dominance property. Recall that f is the objective function introduced on page 41.

Definition 5.2

Let $(E, T) \in \vec{\mathcal{P}}_2(J)$.

(i) (E, T) *is an* ***insert local optimum*** *if* $\begin{cases} \forall v \in T,\ f(E,T) \leqslant f\big(E \cup \{v\},\ T \setminus \{v\}\big), \\ \forall u \in E,\ f(E,T) \leqslant f\big(E \setminus \{u\},\ T \cup \{u\}\big). \end{cases}$

otherwise (E, T) *is* ***insert-dominated***.

(ii) (E, T) *is a* ***swap local optimum*** *if* $\forall u \in E,\ \forall v \in T,\ f(E,T) \leqslant f\big(E \setminus \{u\} \cup \{v\},\ T \setminus \{v\} \cup \{u\}\big)$,

otherwise (E, T) *is* ***swap-dominated***.

Property 5.3

The set of insert locally optimal partitions, as well as the set of swap locally optimal partitions, are strictly dominant when minimizing f *over* $\vec{\mathcal{P}}_2(J)$.

In the next section, we provide linear inequalities translating the insert local optimum dominance property. The inequalities related to swap operations will be provided later (Cf. Section 5.4).

5.2 Linear inequalities for the insert dominance property

• *Some notations and definitions to describe V-shaped d-blocks*

In order to provide an expression of the penalty variation induced by such an operation, it is convenient to choose a specific V-shaped d-block as the representative of an equivalence class. More precisely, the chosen representative will depend on the inserted task. Let us introduce some notations, related to a given task $u \in J$. To describe the early side of a V-shaped d-block with regard to u, the set of remaining tasks $J \backslash \{u\}$ can be split according to their α-ratio into two subsets $A(u)$ and $\bar{A}(u)$ defined as follows.

$$\boldsymbol{A}(u) = \left\{ i \in J \;\middle|\; \frac{\alpha_i}{p_i} > \frac{\alpha_u}{p_u} \right\} \quad \text{and} \quad \bar{\boldsymbol{A}}(u) = \left\{ i \in J \backslash \{u\} \;\middle|\; \frac{\alpha_i}{p_i} \leqslant \frac{\alpha_u}{p_u} \right\}$$

Similarly, we introduce the two following subsets to describe the tardy side.

$$\boldsymbol{B}(u) = \left\{ i \in J \;\middle|\; \frac{\beta_i}{p_i} > \frac{\beta_u}{p_u} \right\} \quad \text{and} \quad \bar{\boldsymbol{B}}(u) = \left\{ i \in J \backslash \{u\} \;\middle|\; \frac{\beta_i}{p_i} \leqslant \frac{\beta_u}{p_u} \right\}$$

Note that if u is early in a V-shaped d-block, early tasks belonging to $A(u)$, *i.e.* tasks of $A(u) \cap E$, are necessarily scheduled after u, because of their α-ratio. Conversely, an early task of $\bar{A}(u)$ is not necessarily scheduled before u, when its α-ratio is the same as the one of u. However, we will consider a representative where u is placed after all the early tasks of $\bar{A}(u)$, that is as tardy as possible according to its α-ratio. Similarly, in case where u is tardy, we will consider a representative where u is scheduled before all the tardy tasks of $\bar{B}(u)$. Such a representative will be called a **u-canonical** V-shaped d-block. Figure 5.2 (resp. Figure 5.3) represents the structure of a u-canonical V-shaped d-block in which u is early (resp. tardy).

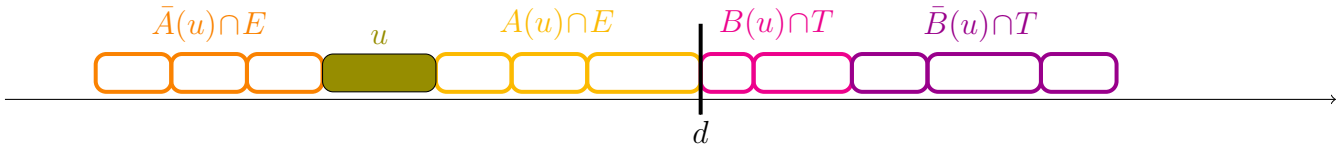


Figure 5.2: Structure of a V-shaped d-block according to a task $u \in E$

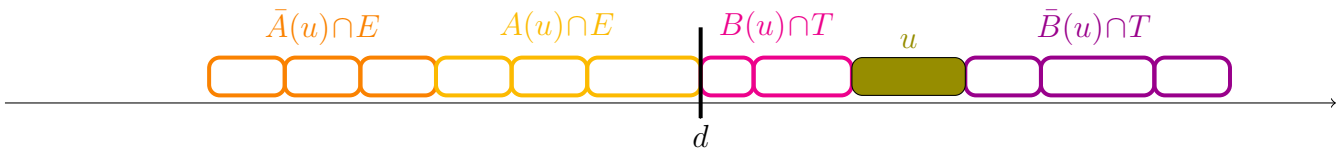


Figure 5.3: Structure of a V-shaped d-block according to a task $u \in T$

• *Penalty variation induced by an insert operation*

Given $u \in J$, let (E, T) be a partition such that $u \in E$. We aim to express the variation of f induced by the insertion of u into T, *i.e.* between (E, T) and $(E \backslash \{u\}, T \cup \{u\})$. To this end, we consider a u-canonical representative $\mathcal{S}$ of (E, T), and the V-shaped d-block $\mathcal{S}'$ obtained from $\mathcal{S}$ by inserting u in T, as early as possible, that is just after tardy tasks of $B(u)$. Note that $\mathcal{S}'$ is thus a u-canonical representative of $\big(E \backslash \{u\}, T \cup \{u\}\big)$. Let (e, t) (resp. (e', t')) denote the earliness and tardiness vector of tasks in $\mathcal{S}$ (resp. $\mathcal{S}'$).

As we can observe in Figure 5.4, early tasks of $A(u)$ and tardy tasks of $B(u)$ are identically scheduled in $\mathcal{S}$ and $\mathcal{S}'$. The penalty variation induced by the insertion of u is then only induced by the move of u and by the right-shifting of early tasks of $\bar{A}(u)$ and tardy tasks of $\bar{B}(u)$.

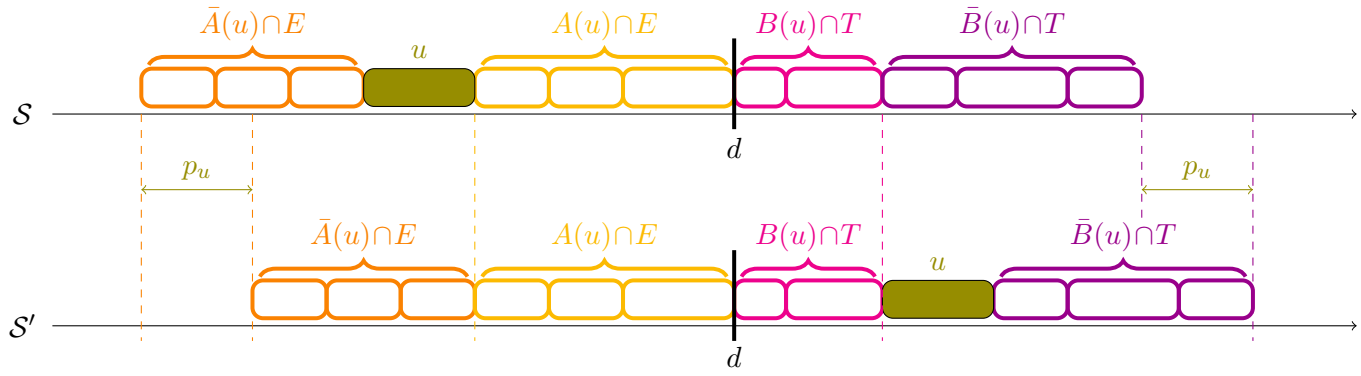


Figure 5.4: Illustration of the insert operation of an early task u on the tardy side

Each task $j \in \bar{A}(u)\cap E$ of $\mathcal{S}$ is shifted p_u time units later in $\mathcal{S}'$, while staying early. We then have $e'_j = e_j - p_u$ and $t'_j = t_j = 0$. The earliness penalty of each task j in $\mathcal{S}'$ is therefore $\alpha_j\, e'_j = \alpha_j\, e_j - \alpha_j\, p_u$, which represents a reduction of $\alpha_j\, p_u$ compared to its earliness penalty in $\mathcal{S}$. Summing up over $\bar{A}(u)\cap E$, we obtain a reduction of the earliness penalties of $p_u\, \alpha\Big(\bar{A}(u)\cap E\Big)$. Similarly, the right-shifting of tasks in $\bar{B}(u)\cap T$ induces an increase of the tardiness penalties of $p_u\, \beta\Big(\bar{B}(u)\cap T\Big)$, since for each $j \in \bar{B}(u)\cap T$ we have $\beta_j\, t'_j = \beta_j\, (t_j + p_u)$.

Moreover, since $e_u = p\Big(A(u)\cap E\Big)$, removing u from the early side induces a reduction of the earliness penalty of $\alpha_u\, p\Big(A(u)\cap E\Big)$. Similarly, introducing u on the tardy side induces an increase of the tardiness penalty of $\beta_u\, t'_u = \beta_u \left(p\Big(B(u)\cap T\Big) + p_u\right)$.

Finally, the penalty variation between $\mathcal{S}$ and $\mathcal{S}'$, is given by the following expression.

$$\boldsymbol{\Delta_u(E,T)} = -\alpha_u\, p\Big(A(u)\cap E\Big) + \beta_u \left(p\Big(B(u)\cap T\Big) + p_u\right) + p_u \left(\beta\Big(\bar{B}(u)\cap T\Big) - \alpha\Big(\bar{A}(u)\cap E\Big)\right)$$

Since $\mathcal{S}$ and $\mathcal{S}'$ are representative of (E,T) and $(E\backslash\{u\}, T\cup\{u\})$ respectively, $\Delta_u(E,T)$ is the variation of f induced by the insertion of u in T. Property 5.4.*(i)* follows.

The insert operation of the tardy task u on the early side applied on schedule $\mathcal{S}'$ provides schedule $\mathcal{S}$. Note that this statement requires that $\mathcal{S}$ is u-canonical. This observation allows to establish that the penalty variation induced by inserting u on the early side in $\mathcal{S}'$ is simply $-\Delta_u(E,T)$, and results in Property 5.4.*(ii)*.

Property 5.4

Let (E,T) be a partition.
(i) For any $u \in E$, $f(E\backslash\{u\}, T\cup\{u\}) = f(E,T) + \Delta_u(E,T)$.
(ii) For any $v \in T$, $f(E\cup\{v\}, T\backslash\{v\}) = f(E,T) - \Delta_v(E,T)$.

Let us introduce, for a given task $u \in J$, the two following constraints[1].

$$u \in E \Rightarrow \Delta_u(E,T) \geqslant 0 \qquad (\hat{I}_u)$$
$$u \in T \Rightarrow \Delta_u(E,T) \leqslant 0 \qquad (\hat{I'}_u)$$

Thanks to Property 5.4, $(\hat{I}_u)$ (resp. $(\hat{I'}_u)$) discards every partition where u is early (resp. tardy) that would have a lower penalty if u was tardy (resp. early). Note that only considering constraints $(\hat{I}_u)$ and $(\hat{I'}_u)$ for a given u is not sufficient to discard all insert-dominated partitions. Indeed, these constraints do not take into account the whole insert-neighborhood of the solutions as each one is only

[1]These constraints are labeled by I as insert, not I as inequality.

compared to its neighbor obtained by an insert operation on task u. It is thus needed to consider constraints $(\hat{I}_u)$ and $(\hat{I'}_u)$ for every $u \in J$ to translate the dominance of the insert local optima.

Moreover, the constraint $\Delta_u(E,T) \geqslant 0$ (resp. $\Delta_u(E,T) \leqslant 0$) might not be satisfied by an optimal solution where u is tardy (resp. early). Therefore, constraints $(\hat{I}_u)$ and $(\hat{I'}_u)$ are not simply inequalities, but conditional statements whose consequent[2] is an inequality. The end of this section is dedicated to using δ variables in order to provide linear inequalities equivalent to constraints $(\hat{I}_u)$ and $(\hat{I'}_u)$.

• *Linear inequalities translating constraints* $(\hat{I}_u)$ *and* $(\hat{I'}_u)$ *for any* $u \in J$

Let $u \in J$. If $\delta \in \{0,1\}^J$ encodes a partition (E,T), the penalty variation $\Delta_u(E,T)$ can be expressed linearly from δ as follows.

$$\boldsymbol{\Delta_u(\delta)} = -\alpha_u \sum_{i \in A(u)} p_i\, \delta_i + \beta_u \Big(\sum_{i \in B(u)} p_i(1-\delta_i) + p_u \Big) + p_u \Big(\sum_{i \in \bar{B}(u)} \beta_i(1-\delta_i) - \sum_{i \in \bar{A}(u)} \alpha_i\, \delta_i \Big)$$

Moreover, (E,T) satisfies $(\hat{I}_u) \Leftrightarrow \Big(u \in E \text{ and } \Delta_u(E,T) \geqslant 0\Big)$ or $u \in T$

$$\Leftrightarrow \Big(\delta_u = 1 \text{ and } \Delta_u(E,T) \geqslant 0\Big) \text{ or } \delta_u = 0$$

To unify these two cases ($\delta_u = 1$ and $\delta_u = 0$) into one inequality, we introduce the following constant which is an upper bound of $-\Delta_u(\delta)$ for any $\delta \in \{0,1\}^J$.

$$\boldsymbol{M_u} = \alpha_u\, p\Big(A(u)\Big) - \beta_u\, p_u + p_u\, \alpha\Big(\bar{A}(u)\Big)$$

Since we have $\Delta_u(\delta) \geqslant -M_u$ for any $\delta \in \{0,1\}^J$, the following inequality is satisfied by every $\delta \in \{0,1\}^J$ such that $(1-\delta_u) = 1$.

$$\Delta_u(\delta) \geqslant -M_u\,(1-\delta_u) \qquad (I_u)$$

Conversely, for every $\delta \in \{0,1\}^J$ such that $(1-\delta_u) = 0$, inequality (I_u) is satisfied if and only if $\Delta_u(\delta) \geqslant 0$. Considering these two cases, $\delta \in \{0,1\}^J$ satisfies (I_u) if and only if the partition encoded by δ satisfies $(\hat{I}_u)$. Property 5.5.*(i)* follows.

Similarly, to translate constraint $(\hat{I'}_u)$, we introduce the following constant which is an upper bound of $-\Delta_u(\delta)$ for any $\delta \in \{0,1\}^J$.

$$\boldsymbol{M'_u} = \beta_u\, p\Big(B(u)\Big) + \beta_u\, p_u + p_u\, \beta\Big(\bar{B}(u)\Big)$$

Since we have $\forall \delta \in \{0,1\}^J$, $-\Delta_u(\delta) \geqslant -M'_u$, the following inequality is satisfied by every $\delta \in \{0,1\}^J$ such that $\delta_u = 1$.

$$-\Delta_u(\delta) \geqslant -M'_u\, \delta_u \qquad (I'_u)$$

Conversely, if $\delta_u = 0$, δ satisfies (I'_u) if and only if $-\Delta_u(\delta) \geqslant 0$. Considering these two cases, $\delta \in \{0,1\}^J$ satisfies (I'_u) if and only if the partition encoded by δ satisfies $(\hat{I}_u)$. Property 5.5.*(ii)* follows.

Property 5.5

Let $\delta \in \{0,1\}^J$ *and let* (E,T) *be the partition encoded by* δ.
For any $u \in J$, *(i)* δ *satisfies inequality* (I_u) ***if and only if*** (E,T) *satisfies constraint* $(\hat{I}_u)$,
(ii) δ *satisfies inequality* (I'_u) ***if and only if*** (E,T) *satisfies constraint* $(\hat{I'}_u)$.

In the sequel, **insert inequalities** will refer to inequalities (I_u) and (I'_u) for all the tasks $u \in J$.

[2]For a conditional statement $p \Rightarrow q$, the consequent is q.

5.3 General framework to produce dominance inequalities from a set of operations

The aim of this section is to generalize this approach by extracting from the previous section some general ideas. Therefore, we first give a high-level explanation of the dominance inequalities based on operations, and explain how it differs from the standard local search framework. Subsequently, we give a property that enables to create dominance inequalities associated with a set of operations. This property is then applied to the swap operations. At the end of the section, we additionally introduce some definitions to qualify dominance inequalities in general.

5.3.1 The solution-centered and the operation-centered points of view

Property 5.3 is based on a solution-centered point of view as all the neighbors of one given solution are taken into account. Conversely, constraints $(\hat{I}_u)$ and $(\hat{I'}_u)$ are based on an operation-centered point of view as only one neighbor of each solution — the one obtained by a given operation — is taken into account. Figures 5.5 and 5.6 illustrate these two points of view on the same set of solutions $\{A, B, \ldots, N\}$ represented by some blue points. We consider a given set of operations and the associated neighborhood function. An arrow is drawn from a solution X to a solution Y if X is compared to Y in order to determine whether X is dominated by Y.

In Figure 5.5, we focus on one solution J, which is compared to all the solutions obtained by applying to J an operation defined on J. Since J is compared to all its neighbors, *i.e.* $\{C, F, G, I, M, N\}$, one can determine if J is locally optimal.

In Figure 5.6, we focus on one operation. All the solutions where this operation can be applied are compared to the obtained neighbors. Solutions A, B, C, G and N are not compared to other solutions, since the considered operation cannot be applied on these solutions. Conversely, since solutions D, E, F, H, I, J, K, L and M are compared to one of their neighbors, they might be discarded. For example if E is better than H, H can be discarded, no matter whether H is better than K. However, we cannot say that the non-discarded solutions are locally optimal, since only one neighbor is taken into account. To this end, all the operations must be considered.

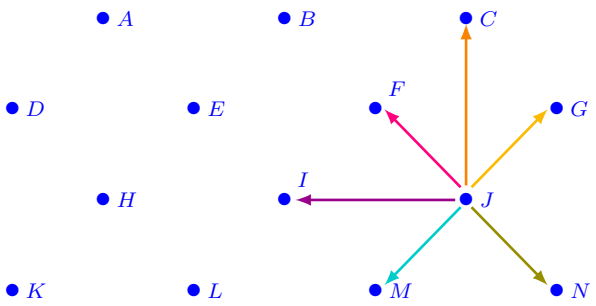


Figure 5.5: Illustration of the *solution-centered* point of view

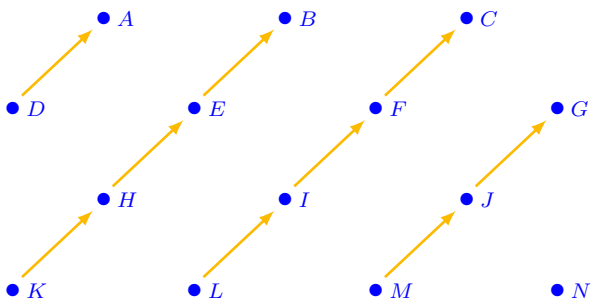


Figure 5.6: Illustration of the *operation-centered* point of view

In the sequel, we focus on the operation-centered point of view. Indeed, we propose a way to obtain, for a given operation and under some assumptions, a linear inequality that cut off all the solutions that can be discarded by comparison with the neighbor obtained by this operation.

5.3.2 A way to obtain a dominance inequality from an operation

Let us consider the minimization problem $\min_{S\in\mathscr{S}} f^{obj}(S)$, where $\mathscr{S}$ is a finite subset of $\mathbb{R}^n$.

Property 5.6

Let θ be an operation on $\mathscr{S}$, i.e. $\theta\in\mathcal{F}(\mathscr{S}',\mathscr{S})$, where $\mathscr{S}'\subseteq\mathscr{S}$ denotes its domain of definition. Let $\mathcal{N}^\theta$ be the related neighborhood function, i.e. $\mathcal{N}^\theta(S)=\{\theta(S)\}$ if $S\in\mathscr{S}'$, $\mathcal{N}^\theta(S)=\emptyset$ otherwise.

***If** there exist:*

$\rightarrow$ Π a linear function from $\mathbb{R}^n$ to $\mathbb{R}$ such that $\Pi(\mathscr{S})\subseteq\mathbb{N}$ and $\forall S\in\mathscr{S}$, $\Pi(S)=0\Leftrightarrow S\in\mathscr{S}'$

$\rightarrow$ Δ a linear function from $\mathbb{R}^n$ to $\mathbb{R}$ such that $\forall S\in\mathscr{S}'$, $\Delta(S)=f^{obj}\Big(\theta(S)\Big)-f^{obj}(\mathcal{S})$

***then** for $\widehat{m}\in\mathbb{R}$ an upper bound of $\{-\Delta(S)\,|\,S\in\mathscr{S}\}$ and $M=\begin{cases}\frac{\widehat{m}}{\max\Pi(\mathscr{S})} & \text{if } \widehat{m}<0\\ \widehat{m} & \text{otherwise}\end{cases}$*

the linear inequality $\boxed{\Delta(S)\geqslant -M\,\Pi(S)}$ translates the $\mathcal{N}^\theta$-dominance.

Proof: Let us assume that such functions Π and Δ exist. Since $\mathscr{S}$ is a finite set, the set $\{-\Delta(S)\,|\,S\in\mathscr{S}\}$ is bounded. Let us consider $\widehat{m}$ an upper bound of this set, and then set M accordingly. Let $S\in\mathscr{S}$.

If $S\notin\mathscr{S}'$, then $\Pi(S)\geqslant 1$ by definition. If $\widehat{m}\geqslant 0$, then we have $M\,\Pi(S)=\widehat{m}\,\Pi(S)\geqslant\widehat{m}$, otherwise we have $M=\frac{\widehat{m}}{\max\Pi(\mathscr{S})}\leqslant 0$, and then $M\,\Pi(S)\geqslant M\,\max\Pi(\mathscr{S})=\widehat{m}$. In both cases, we have $M\,\Pi(S)\geqslant\widehat{m}$. By definition of $\widehat{m}$, we deduce that $M\,\Pi(S)\geqslant-\Delta(S)$, that is $-M\,\Pi(S)\leqslant\Delta(S)$. In other words, S satisfies the inequality.

If $S\in\mathscr{S}'$, $\Pi(S)=0$ by definition, then $\Delta(S)\geqslant -M\,\Pi(S)\Leftrightarrow\Delta(S)\geqslant 0\Leftrightarrow f^{obj}(\theta(S))-f^{obj}(S)\geqslant 0$. In other words, S satisfies the inequality if and only if S is θ-dominant. □

In the case of a maximization problem, under the same assumptions, the $\mathcal{N}^\theta$-dominance is translated by the inequality $\boxed{\Delta(S)\leqslant M\,\Pi(S)}$ where M is defined from an upper bound $\overset{\vee}{m}$ of $\{\Delta(S)\,|\,S\in\mathscr{S}\}$ instead of from $\widehat{m}$.

Note that these properties can also be used for a set of operations Θ instead of for a single operation. Indeed, if I_θ denotes the inequality build according to Property 5.6 for the operation $\theta\in\Theta$, then the family $(I_\theta)_{\theta\in\Theta}$ translates the $\mathcal{N}^\Theta$-dominance property, where $\mathcal{N}^\Theta$ is the neighborhood function associated with the operation set Θ, that is $\mathcal{N}^\Theta=S\mapsto\{\theta(S)\,|\,\theta\in\Theta\text{ s.t. }s\in\text{domain}(\theta)\}$.

We apply Property 5.6 to the swap operation for UCDDP in the next sub-section. In Chapter 7, the maximization version of this property will be used for other problems.

5.3.3 Qualifying for dominance inequalities in a general framework

In order to be able to qualify dominance inequalities, we introduce some definitions, adapted from the classical ones. Let us consider the linear formulation of a generic minimization problem, defined as follows. (Of course, the following definitions can be transposed to a maximization problem.)

$$F : \min_{Y \in \widetilde{P} \cap \mathbb{Z}^{\dim}} f^{obj}(Y)$$

where:

- $\mathscr{S}$ a finite set of solutions in $\mathbb{R}^{\dim}$, with $\dim \in \mathbb{N}^*$
- $\boldsymbol{f^{obj}}$ a linear objective function defined on $\mathbb{R}^{\dim}$, *i.e.* $f^{obj} \in \mathcal{F}(\mathbb{R}^{\dim}, \mathbb{R})$
- $\mathscr{S}^*$ the set of the optimal solutions, *i.e.* $\mathscr{S}^* = \arg\min_{S \in \mathscr{S}} f^{obj}(S)$
- $\boldsymbol{P}$ the convex hull of the solution set, *i.e.* $P = \text{conv}(\mathscr{S})$
- $\widetilde{\boldsymbol{P}}$ a polyhedron such that $\widetilde{P} \cap \mathbb{Z}^{\dim} = \mathscr{S}$

In this framework, let us introduce some definitions for an inequality I of $\mathbb{R}^{\dim}$.

-I is **valid** if any solution $S \in \mathscr{S}$ satisfies I.

-I is f^{obj}**-valid** if any optimal solution $S^* \in \mathscr{S}^*$ satisfies I.

-I is a **cut** for F if there exists a point in $\widetilde{P}$ that does not satisfy I.

-I is a f^{obj}**-cut** for F if there exists a minimizer in $\arg\min_{S \in \widetilde{P}} f^{obj}(S)$ that does not satisfy I.

Note that I is valid $\Rightarrow$ I is f^{obj}-valid while conversely I is a f^{obj}-cut $\Rightarrow$ I is a cut. In general, reinforcement inequalities are valid cuts, or valid f^{obj}-cuts, which is better. Dominance inequalities are f^{obj}-valid cuts, indeed, such inequalities cut off some elements in $\mathscr{S}$, namely the dominated solutions.

To improve the linear relaxation value, *i.e.* $\min_{S \in \widetilde{P}} f^{obj}(S)$ all the minimizers of f^{obj} over $\widetilde{P}$ must be removed. If there is only one minimizer, a single f^{obj}-cut can improve the linear relaxation value. In general, there can be several minimizers, which can be cut off by different f^{obj}-cuts. Therefore, we introduce the following definition for a family of linear inequalities of $\mathbb{R}^{\dim}$ denoted by $\mathcal{I}$.

-$\mathcal{I}$ is f^{obj}**-enhancing** (for $\widetilde{P}$) if $\forall S^* \in \arg\min_{\mathcal{S} \in \widetilde{P}} f^{obj}(S)$, $\exists I \in \mathcal{I}$, S^* does not satisfy I.

It can be difficult to imagine that a dominance inequality could be a f^{obj}-cut. How an inequality which cuts integer points with a high value according to the objective function can also cut an optimal fractional point, that is by definition a point with a low value according to the objective function? We propose below an illustration of such an inequality.

5.3.4 An illustration of a generic f^{obj}-cut dominance inequality

We provide an illustration of a generic situation with 2 binary variables. Let us use the notations introduced on the previous page to describe the generic formulation F. In this case, $\dim = 2$ and $P \subseteq [0,1]^2$.

Figure 5.7 represents the polytope $\widetilde{P}$ whose integer points are exactly points of $\mathscr{S}$. The optimization direction is represented by a red arrow, and dotted red lines represent points having the same value according to f^{obj} (like contour lines on a map represent points having the same altitude). The optimal (integer) solution is represented by a blue star, while the fractional optimum is represented by a violet star.

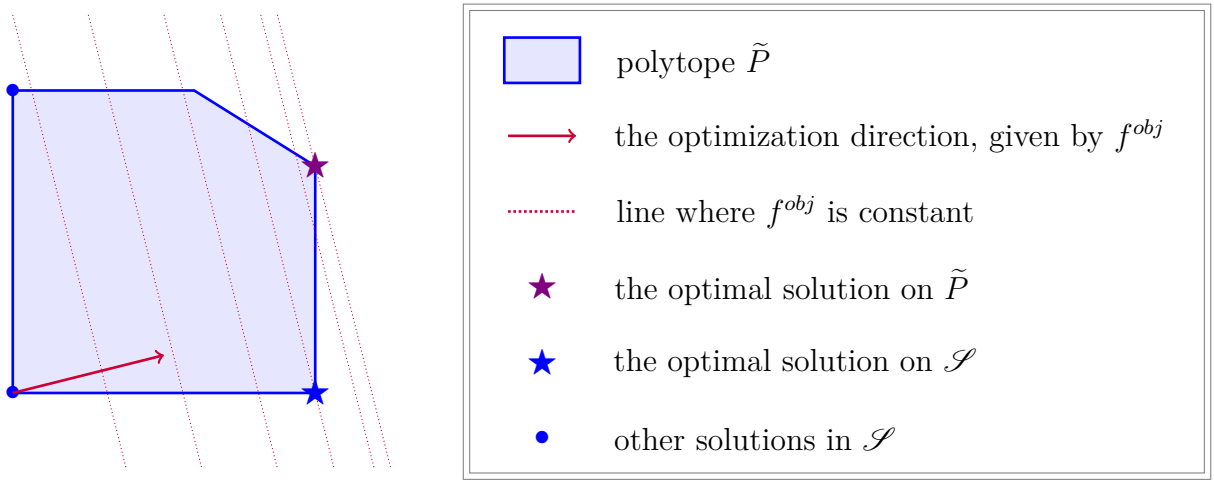


Figure 5.7: Optimal solution in $\mathscr{S}$ (★) and optimal point in $\widetilde{P}$ (★)

Figure 5.8 shows an inequality, drawn in orange, that cuts an integer point which is dominated, and also cuts the fractional optimum. The new fractional optimum, represented by a turquoise star, offers a better linear relaxation value.

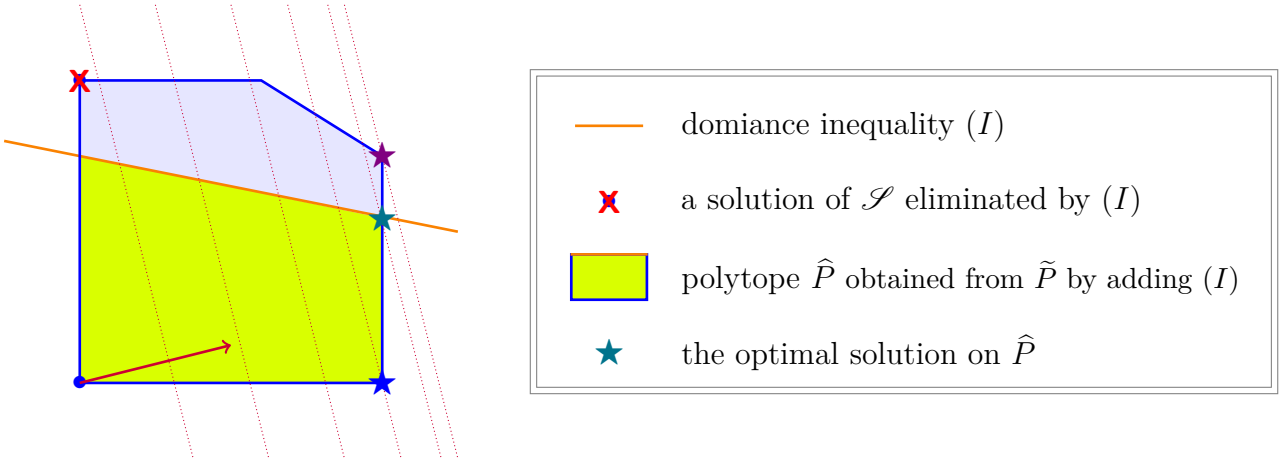


Figure 5.8: Optimal point in $\widehat{P}$ (★) obtained by adding a dominance inequality

.

5.4 Application for swap operations

Let $(u,v) \in J^2$ such that $u \neq v$. We have defined, on page 152, the swap operation as an operation on *ordered bi-partitions*. However, to stick to the Property 5.6 framework, we need to consider an operation on *vectors*. Therefore we introduce the following operation[3].

$$\theta_{u,v} = \begin{pmatrix} \mathscr{S}_{u,v} & \longrightarrow S^n_{\delta,X} \subseteq \mathbb{R}^V \times \mathbb{R}^E \\ (\delta, X) & \longmapsto \psi^{\{u,v\}}(\delta, X) \end{pmatrix} \quad \text{where} \quad \mathscr{S}_{u,v} = \Big\{ (\delta, X) \in S^n_{\delta,X} \;\Big|\; \delta_u = 1 \text{ and } \delta_v = 0 \Big\}$$

Note that $\mathscr{S}_{u,v}$ is exactly the set of vectors that encode a partition (E,T) such that $u \in E$ and $v \in T$. In order to apply Property 5.6, we have to identify suitable functions Δ and Π, before defining a constant M, and finally obtain a dominance inequality.

• *Identifying function* Π

To identify the domain of $\theta_{u,v}$, *i.e.* $\mathscr{S}_{u,v}$, we introduce the following linear function.

$$\Pi_{u,v} = \begin{pmatrix} \mathbb{R}^V \times \mathbb{R}^E & \longrightarrow \mathbb{R} \\ (\delta, X) & \longmapsto (1-\delta_u) + \delta_v \end{pmatrix}$$

For any $(\delta, X) \in S^n_{\delta,X}$, we have $\Pi_{u,v}(\delta, X) = 0 \Leftrightarrow \underbrace{(1-\delta_u)}_{\geqslant 0} + \underbrace{\delta_v}_{\geqslant 0} = 0 \Leftrightarrow \begin{cases} (1-\delta_u) = 0 \\ \delta_v = 0 \end{cases} \Leftrightarrow (\delta, X) \in \mathscr{S}_{u,v}$.
Therefore, $\Pi_{u,v}$ is suitable with Property 5.6.
Moreover, one can already note that $\max \Pi_{u,v} = 2$, which will be useful subsequently to define $M_{u,v}$.

• *Identifying function* Δ

To identify $\Delta_{u,v}$, we have to express the variation of $h_{\alpha,\beta}$ — the objective function — between a vector (δ, X) in $\mathscr{S}_{u,v}$ and its image $\theta_{u,v}(\delta, X)$. By construction, that corresponds to the variation of f between a partition (E,T) such that $u \in E$ and $v \in T$, and its image $(E \setminus \{u\} \cup \{v\}, T \setminus \{v\} \cup \{u\})$, then it also corresponds to the penalty variation between a V-shaped d-block $\mathcal{S}$ in which u is early and v is tardy and its image $\text{swap}_{u,v}(\mathcal{S})$. As we find it easier, we choose to express the penalty variation between V-shaped d-blocks, and even between specific V-shaped d-blocks as done for the insert operation.

Let (E,T) be a partition such that $u \in E$ and $v \in T$, and $\mathcal{S}$ be a u and v-canonical representative of (E,T). We denote by $\mathcal{S}'$ the schedule obtained from $\mathcal{S}$ by swapping u and v so that $\mathcal{S}'$ is also both u and v-canonical, that is by scheduling u after all the early tasks having the same α-ratio α_u/p_u and v before all the tardy tasks having the same β-ratio β_v/p_v. $\mathcal{S}'$ is a representative of $\Big(E \setminus \{u\} \cup \{v\}, T \setminus \{v\} \cup \{u\}\Big)$. Let (e,t) (resp. (e',t')) denote the earliness and tardiness vector of tasks in $\mathcal{S}$ (resp. $\mathcal{S}'$).

If $\frac{\alpha_v}{p_v} < \frac{\alpha_u}{p_u}$, all the early tasks of $\bar{A}(v)$ are scheduled before u in $\mathcal{S}$, since the V-shaped property holds, but the early tasks of $A(v) \setminus \{u\}$ can be scheduled before or after u. This case is illustrated in Figure 5.9. Conversely, if $\frac{\alpha_v}{p_v} \geqslant \frac{\alpha_u}{p_u}$, all the early tasks of $A(v)$ are scheduled after u in $\mathcal{S}$, but those of $A(v) \setminus \{u\}$ can be scheduled before or after u. This case is illustrated in Figure 5.10.

Similarly, we can distinguish two cases depending on the relative order of the β-ratios of u and v, as illustrated in Figures 5.11 and 5.12.

[3]Function $\psi^{\{u,v\}}$ used to define $\theta_{u,v}$ is the function ψ^M defined on page 108 for $M = \{u,v\}$.

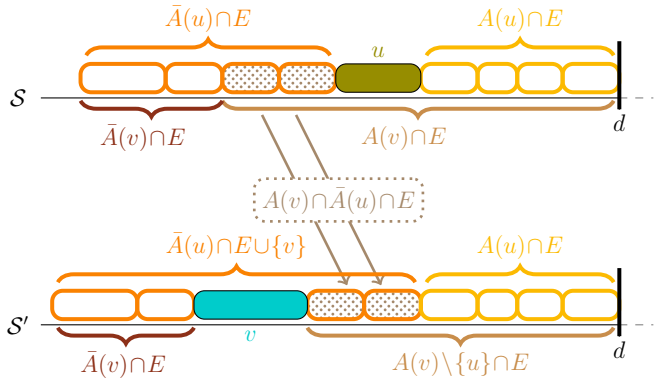


Figure 5.9: Early side variation induced by swapping u and v when $\frac{\alpha_v}{p_v} < \frac{\alpha_u}{p_u}$

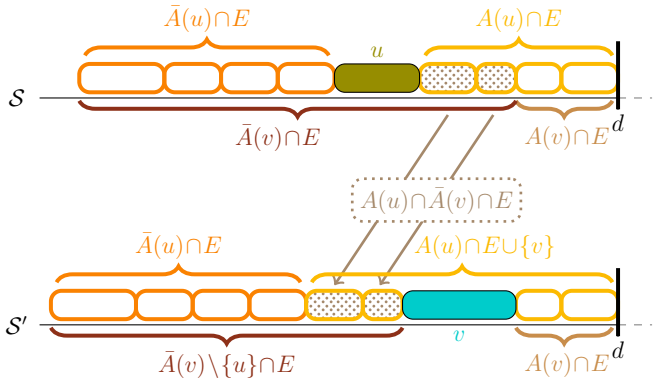


Figure 5.10: Early side variation induced by swapping u and v when $\frac{\alpha_v}{p_v} \geqslant \frac{\alpha_u}{p_u}$

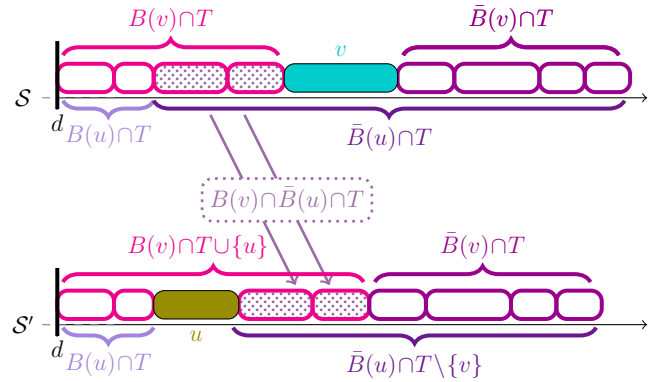


Figure 5.11: Tardy side variation induced by swapping u and v when $\frac{\beta_v}{p_v} \leqslant \frac{\beta_u}{p_u}$

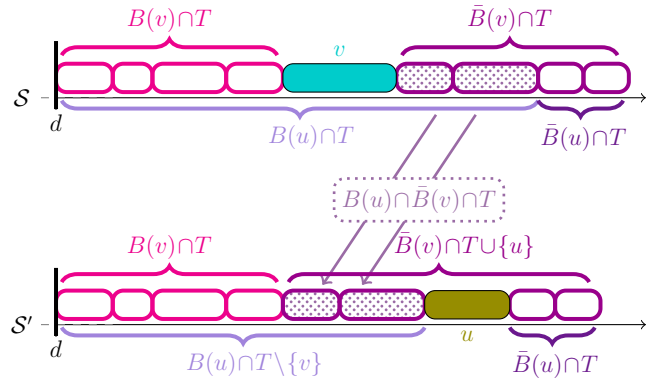


Figure 5.12: Tardy side variation induced by swapping u and v when $\frac{\beta_v}{p_v} > \frac{\beta_u}{p_u}$

As shown in Figures 5.9 and 5.10, the earliness of u in $\mathcal{S}$ is $e_u = p\big(A(u)\cap E\big)$, while the tardiness of u in $\mathcal{S}'$ is $e'_u = p\big(B(u)\cap T\big) + p_u$ (*Cf.* Figures 5.11 and 5.12). Note that v is removed from $B(u)$ since v is not tardy in $\mathcal{S}'$, and therefore cannot contribute to the tardiness of u. Similarly, we have $t_v = p\big(B(u)\cap T\big) + p_v$ and $e'_v = p\big(A(v)\setminus\{u\}\cap E\big)$ since u is not early in $\mathcal{S}'$.

Moreover, the tasks of $A(u)\cap E$ are identically scheduled in $\mathcal{S}$ and $\mathcal{S}'$ only if $\frac{\alpha_v}{p_v} < \frac{\alpha_u}{p_u}$. In this case, tasks of $\bar{A}(u)\cap E$ are not consecutive in $\mathcal{S}'$ since v separates them into two blocks: tasks of $\bar{A}(v)\cap E$ which have been left-shifted by $p_v - p_u$ time units, and tasks of $A(v)\cap\bar{A}(u)\cap E$ which have been right-shifted by p_u time units (*Cf.* Figure 5.9).

In the opposite case, *i.e.* if $\frac{\alpha_v}{p_v} \geqslant \frac{\alpha_u}{p_u}$, tasks of $A(u)\cap E$ are not consecutive in $\mathcal{S}'$. Indeed, v separates them into two blocks: tasks of $A(v)\cap E$ which are identically scheduled in $\mathcal{S}$ and $\mathcal{S}'$; and tasks of $A(u)\cap\bar{A}(v)\cap E$ which are left-shifted by p_v time units. Moreover, in that case tasks of $A(v)\cap E$ are left-shifted by $p_v - p_u$ time units (*Cf.* Figure 5.10).

> *NB:* In the two previous paragraphs, as in Figures 5.9 to 5.12, we assumed that $p_v - p_u \geqslant 0$. In the contrary case, *i.e.* if $p_v - p_u < 0$, tasks are not left-shifted by $p_v - p_u$ time units but rather right-shifted by $p_u - p_v$ time units.

From these observations, we can express the earliness penalty variation of all the tasks in $E\backslash\{u\}$ by expressing, in each case, the penalty variation induced by a block shifting as described for the insert operation. The same method can be applied for the tasks of $T\backslash\{v\}$. The reader can refer to Figures 5.11 and 5.12 for illustration. Finally, the penalty variation between $\mathcal{S}$ and $\mathcal{S}'$ — which is equal to $f(E\backslash\{u\}\cup\{v\}, T\backslash\{v\}\cup\{u\}) - f(E,T)$ by construction — is given by the following expression.

$$\begin{aligned}
\boldsymbol{\Delta_{u,v}(E,T)} = &- \alpha_u\, p\Big(\underline{A(u)\cap E}\Big) + \beta_u \left(p\Big(\underline{B(u)\cap T}\Big) + p_u\right) \\
&- \beta_v \left(p\Big(\underline{B(u)\cap T}\Big) + p_v\right) + \alpha_v\, p\Big(\underline{A(v)\backslash\{u\}\cap E}\Big) \\
&+ \begin{cases} (-p_u+p_v)\,\alpha\Big(\underline{\bar{A}(v)\cap E}\Big) - p_u\,\alpha\Big(\underline{A(v)\cap\bar{A}(u)\cap E}\Big) & \text{if } \dfrac{\alpha_v}{p_v} < \dfrac{\alpha_u}{p_u} \\ (-p_u+p_v)\,\alpha\Big(\underline{\bar{A}(u)\cap E}\Big) + p_v\,\alpha\Big(\underline{A(u)\cap\bar{A}(v)\cap E}\Big) & \text{otherwise} \end{cases} \\
&+ \begin{cases} (p_u-p_v)\,\beta\Big(\underline{\bar{B}(v)\cap T}\Big) + p_u\,\beta\Big(\underline{B(v)\cap\bar{B}(u)\cap T}\Big) & \text{if } \dfrac{\beta_v}{p_v} \leqslant \dfrac{\beta_u}{p_u} \\ (p_u-p_v)\,\beta\Big(\underline{\bar{B}(u)\cap T}\Big) - p_v\,\beta\Big(\underline{B(u)\cap\bar{B}(v)\cap T}\Big) & \text{otherwise} \end{cases}
\end{aligned}$$

Using δ variables, this variation can be written as follows.

$$\begin{aligned}
\boldsymbol{\Delta_{u,v}(\delta)} \;=\; &- \alpha_u \sum_{i\in A(u)} p_i\,\delta_i + \beta_u \sum_{i\in B(u)\backslash\{v\}} p_i\,(1-\delta_i) + \beta_u p_u - \beta_v \sum_{i\in B(v)} p_i\,(1-\delta_i) - \beta_v p_v + \alpha_v \sum_{i\in A(v)\backslash\{u\}} p_i\,\delta_i \\
&+ \begin{cases} (-p_u+p_v) \displaystyle\sum_{i\in\bar{A}(v)} \alpha_i\,\delta_i \;-\; p_u \sum_{i\in A(v)\cap\bar{A}(u)} \alpha_i\,\delta_i & \text{if } \dfrac{\alpha_v}{p_v} < \dfrac{\alpha_u}{p_u} \\ (-p_u+p_v) \displaystyle\sum_{i\in\bar{A}(u)} \alpha_i\,\delta_i \;+\; p_v \sum_{i\in A(u)\cap\bar{A}(v)} \alpha_i\,\delta_i & \text{otherwise} \end{cases} \\
&+ \begin{cases} (p_u-p_v) \displaystyle\sum_{i\in\bar{B}(v)} \beta_i\,(1-\delta_i) \;+\; p_u \sum_{i\in B(v)\cap\bar{B}(u)} \beta_i\,(1-\delta_i) & \text{if } \dfrac{\beta_v}{p_v} \leqslant \dfrac{\beta_u}{p_u} \\ (p_u-p_v) \displaystyle\sum_{i\in\bar{B}(u)} \beta_i\,(1-\delta_i) \;-\; p_v \sum_{i\in B(u)\cap\bar{B}(v)} \beta_i\,(1-\delta_i) & \text{otherwise} \end{cases}
\end{aligned}$$

• *Defining constant M*
We introduce $\widehat{m}_{u,v}$ an upper bound of $\{-\Delta_{u,v}(\delta)\,|\,(\delta,X)\in\mathscr{S}_{u,v}\}$ to define $\boldsymbol{M_{u,v}} = \begin{cases} \widehat{m}_{u,v} & \text{if } \widehat{m}_{u,v} \geqslant 0 \\ \frac{\widehat{m}_{u,v}}{2} & \text{otherwise} \end{cases}$

$$\begin{aligned}
\boldsymbol{\widehat{m}_{u,v}} \;=\; &\alpha_u\, p\Big(A(u)\Big) - \beta_u\, p_u + \beta_v\, p\Big(B(v)\Big) + \beta_v p_v \\
&+ \begin{cases} [p_u-p_v]^+\,\alpha\Big(\bar{A}(v)\Big) + p_u\,\alpha\Big(A(v)\cap\bar{A}(u)\Big) & \text{if } \dfrac{\alpha_v}{p_v} < \dfrac{\alpha_u}{p_u} \\ [p_u-p_v]^+\,\alpha\Big(\bar{A}(u)\Big) & \text{if } \dfrac{\alpha_v}{p_v} \geqslant \dfrac{\alpha_u}{p_u} \end{cases} \\
&+ \begin{cases} [p_v-p_u]^+\,\beta\Big(\bar{B}(v)\Big) & \text{if } \dfrac{\beta_v}{p_v} \leqslant \dfrac{\beta_u}{p_u} \\ [p_v-p_u]^+\,\beta\Big(\bar{B}(u)\Big) + p_v\,\beta\Big(B(u)\cap\bar{B}(v)\Big) & \text{if } \dfrac{\beta_v}{p_v} > \dfrac{\beta_u}{p_u} \end{cases}
\end{aligned}$$

• *Finally writing a dominance inequality*

Finally, we obtain the following inequality, which will be called **swap inequality** in the sequel.

$$\Delta_{u,v}(\delta) \geqslant -M_{u,v}\Big(\delta_v + (1-\delta_u)\Big) \qquad (S_{u,v})$$

The following property is a direct corollary of Property 5.6.

Property 5.7

Let $\delta \in \{0,1\}^J$ and let (E,T) be the partition encoded by δ. For any $(u,v) \in J^2$ such that $u \neq v$, δ satisfies $(S_{u,v})$ ***if and only if*** *$(u,v) \notin E \times T$ or $f(E \setminus \{u\} \cup \{v\}, T \setminus \{v\} \cup \{u\}) \geqslant f(E,T)$.*

5.5 Additional properties on insert and swap inequalities

This section aims to present additional results on the two families of dominance inequalities provided in this chapter. More precisely, we try to answer the following questions.

- Why considering swap inequalities in addition to insert inequalities, knowing that a swap operation is equivalent to two insert operations? Are swap inequalities not redundant with insert inequalities?
- Why not considering dominance inequalities corresponding to double insert operations? A double insert operation means an operation that consists in simultaneously inserting two early tasks on the tardy side, or conversely two tardy tasks on the early side.
- Are insert and swap inequalities able to improve the linear relaxation value of F_l^2?

5.5.1 Can two insert inequalities replace a swap inequality?

To answer this question on insert and swap inequalities, let us focus on their underlying operations.

Let us set $(u, v) \in J^2$ such that $u \neq v$. Let $(E, T) \in \vec{\mathcal{P}}_2(J)$ such that $u \in E$ and $v \in T$. Figure 5.13 illustrates that applying the swap operation for tasks u and v on (E, T) results in the same partition than applying first the insert operation for task u and then the insert operation for task v (or *vice versa*).

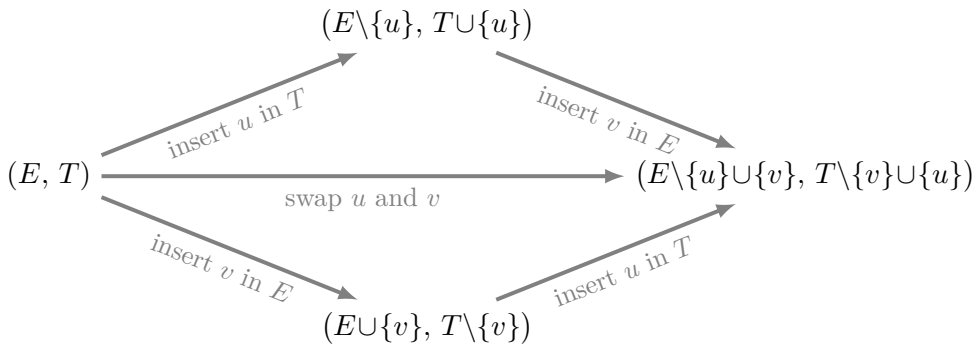


Figure 5.13: Applying insert or swap operations

In other words, a swap operation can be decomposed into two insert operations. Since each dominance inequality cuts an integer point $(\delta, X) \in \{0,1\}^V \times \{0,1\}^E$ if and only if the corresponding operation can improve the encoded partition (*Cf.* Property 6.1), we wonder if the swap operation allows an improvement while each insert operation alone does not.[4]

The answer is yes, as shown by the counter-example below (*Cf.* page 165). Note that, for the sake of simplicity, the counter-example is given on V-shaped d-blocks and not on partitions, which is equivalent.

To conclude, swap inequalities have their own utility: they discard non-optimal solution that insert inequalities do not discard.

[4]This question is not equivalent to the question of inequality redundancy: even if the swap inequality did not cut off more integer points than the insert inequalities, it could cut off more fractional extreme points. However, since the answer is yes, it suffices to answer to the question of inequality redundancy.

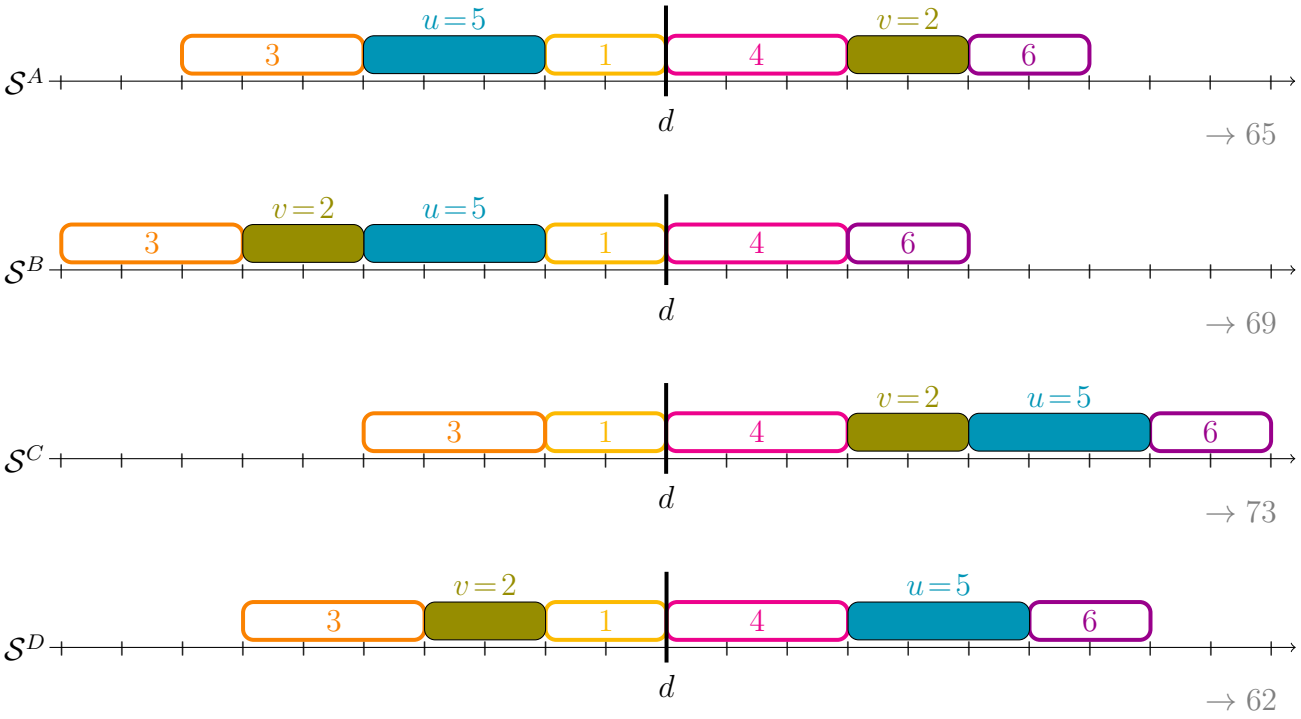
Counter-example 5 : *Schedules that are cut off by a swap inequality are not necessarily also cut off by the two corresponding insert inequalities*

Let us consider the 6-task instance of UCDDP defined by the parameters on the right where α_4, α_6, β_1, and β_3 can be chosen arbitrarily.

i	p_i	α_i	α_i/p_i	β_i	β_i/p_i
1	2	4	2	-	-
2	2	3	$\frac{3}{2}$	3	$\frac{3}{2}$
3	3	3	2	-	-
4	3	-	-	6	2
5	3	5	$\frac{5}{3}$	3	1
6	2	-	-	1	$\frac{1}{2}$

Let us denote by u the task 2, and by v the task 5.

In schedule $\mathcal{S}^A$, u is early and v is tardy. The total penalty of $\mathcal{S}^A$, which is equal to 65, is reduced neither by inserting v on the early side, resulting in $\mathcal{S}^B$ whose total penalty is $69 \geqslant 65$, neither by inserting u on the tardy side, resulting in $\mathcal{S}^C$ whose total penalty is $73 \geqslant 65$. However, swapping tasks u and v results in $\mathcal{S}^D$ whose total penalty is $62 \leqslant 65$.



5.5.2 Could a double insert inequality be useful?

As shown by Counter-example 5, simultaneously swapping two tasks can reduce the penalty even if inserting only one does not. Therefore, one can wonder if simultaneously inserting two early tasks can reduce the penalty when inserting only one does not. If so, we would be interested in inequalities translating the dominance associated with this double insert operation. But the answer to this question is no, as stated in Property 5.8. Since the proof argument is enclosed in the case where the two tasks are consecutive, we only prove Property 5.9. Figure 5.14 illustrates that applying the double insert operation for two early tasks u and u' results in the same partition than applying first the insert operation for task u and then the insert operation for task u' (or *vice versa*).

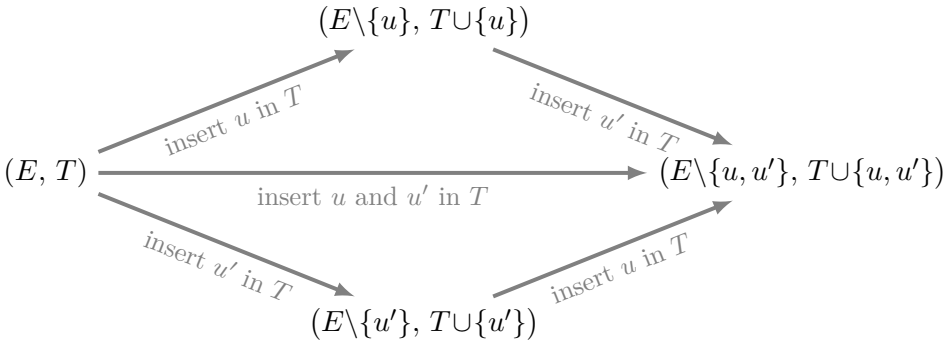


Figure 5.14: Applying insert or double insert operations

Property 5.8

Let $(E,T)\in\vec{\mathcal{P}}_2(J)$. *Let* $(u,u')\in E\times E$ *such that* $u\neq u'$. *We can assume w.l.o.g, that* $\frac{\alpha_{u'}}{p_{u'}}\leqslant\frac{\alpha_u}{p_u}$.

We have $f\Big(E\backslash\{u,u'\},T\cup\{u,u'\}\Big)<f(E,T)\Rightarrow\begin{cases}f\Big(E\backslash\{u\},T\cup\{u\}\Big)<f(E,T)\\ \textit{or}\\ f\Big(E\backslash\{u'\},T\cup\{u'\}\Big)<f(E,T)\end{cases}$

Property 5.9

Let $(E,T)\in\vec{\mathcal{P}}_2(J)$. *Let* $(u,u')\in E\times E$ *such that* $u\neq u'$. *We can assume w.l.o.g, that* $\frac{\alpha_{u'}}{p_{u'}}\leqslant\frac{\alpha_u}{p_u}$.

If $\forall v\in J,\ \frac{\alpha_{u'}}{p_{u'}}<\frac{\alpha_v}{p_v}<\frac{\alpha_u}{p_u}\Rightarrow v\in T$ $(\clubsuit)$,

then $f\Big(E\backslash\{u,u'\},T\cup\{u,u'\}\Big)<f(E,T)\Rightarrow\begin{cases}f\Big(E\backslash\{u\},T\cup\{u\}\Big)<f(E,T)\\ \textit{or}\\ f\Big(E\backslash\{u'\},T\cup\{u'\}\Big)<f(E,T)\end{cases}$

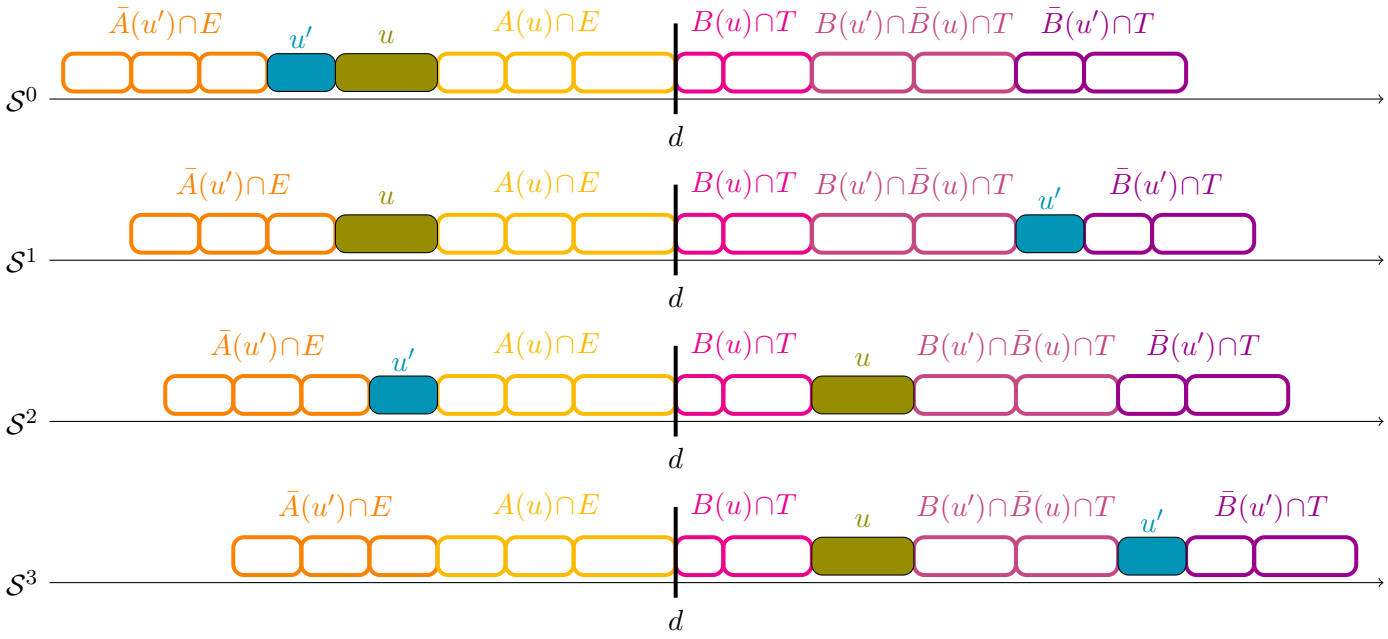
Proof: Let us assume that $\frac{\beta_{u'}}{p_{u'}}\leqslant\frac{\beta_u}{p_u}$, the opposite case is treated similarly.

Thanks to the assumption $(\clubsuit)$, there exists $\mathcal{S}^0$, a u-canonical V-shaped d-block whose partition is (E,T) and in which u and u' are consecutive. We will use $\mathcal{S}^0$ to express the penalty variation induced by different insert operations.

To this end, let us denote by $\mathcal{S}^1$ (resp. $\mathcal{S}^2$) the schedule obtained by inserting u' (resp. u) on the tardy side, *i.e.* $\mathcal{S}^1=\text{insert}_{u'}(\mathcal{S})$ (resp. $\mathcal{S}^2=\text{insert}_u(\mathcal{S})$). Finally, let us consider $\mathcal{S}^3$ the schedule obtained by inserting both u and u' on the tardy side. The shape of these schedules is represented on the top of the next page.

Since each of these four schedules is a V-shaped d-block, their total penalty is given by function f from their early-tardy partition. Therefore, we have the following equalities.

$$(\spadesuit)\begin{cases}\text{penalty}(S^0)=f(E,T)\\ \text{penalty}(S^1)=f(E\backslash\{u'\},T\cup\{u'\})\\ \text{penalty}(S^2)=f(E\backslash\{u\},T\cup\{u\})\\ \text{penalty}(S^3)=f(E\backslash\{u,u'\},T\cup\{u,u'\})\end{cases}$$



The penalty variation between $\mathcal{S}^0$ and $\mathcal{S}^1$, denoted by Δ^1, is only due to the move of u on the tardy side and the right-shifting of tasks in $\bar{A}(u')\cap E$ and $\bar{B}(u')\cap T$. Therefore, this variation can be expressed as follows.

$$\begin{aligned}\Delta^1 &= \text{penalty}(\mathcal{S}^1) - \text{penalty}(\mathcal{S}^0)\\ &= -\alpha_{u'}\left(p(A(u)\cap E) + p_u\right) + \beta_{u'}\left(p(B(u)\cap T)+p(B(u')\cap\bar{B}(u)\cap T)+p_{u'}\right)\\ &\quad - p_{u'}\,\alpha(\bar{A}(u')\cap E) + p_{u'}\,\beta(\bar{B}(u')\cap T)\end{aligned}$$

Similarly, the penalty variation between $\mathcal{S}^0$ and $\mathcal{S}^2$ is denoted by Δ^2 and expressed as follows.

$$\begin{aligned}\Delta^2 &= \text{penalty}(\mathcal{S}^2) - \text{penalty}(\mathcal{S}^0)\\ &= -\alpha_u\,p(A(u)\cap E) + \beta_u\left(p(B(u)\cap T)+p_u\right)\\ &\quad - p_u\left(\alpha(\bar{A}(u')\cap E)+\alpha_{u'}\right) + p_u\left(\beta(B(u')\cap\bar{B}(u)\cap T)+\beta(\bar{B}(u')\cap T)\right)\end{aligned}$$

Finally, the penalty variation between $\mathcal{S}^0$ and $\mathcal{S}^3$, denoted by Δ^3, can be expressed as follows.

$$\begin{aligned}\Delta^3 &= \text{penalty}(\mathcal{S}^3) - \text{penalty}(\mathcal{S}^0)\\ &= -\alpha_u\,p(A(u)\cap E) + \beta_u\left(p(B(u)\cap T)+p_u\right)\\ &\quad - \alpha_{u'}\left(p(A(u)\cap E)+p_u\right) + \beta_{u'}\left(p(B(u)\cap T)+p_u+p(B(u')\cap\bar{B}(u)\cap T)+p_{u'}\right)\\ &\quad - (p_u+p_{u'})\,\alpha(\bar{A}(u')\cap E) + p_u\,\beta(B(u')\cap\bar{B}(u)\cap T) + (p_u+p_{u'})\,\beta(\bar{B}(u')\cap T)\\ &= (\Delta^2 + p_u\,\alpha_{u'}) + \Delta^1 + p_u\,\beta_{u'}\end{aligned}$$

We have $\Delta^3 = \Delta^1+\Delta^2+p_u(\alpha_{u'}+\beta_{u'})$. Since $p_u(\alpha_{u'}+\beta_{u'})\geqslant 0$, $\Delta^3<0 \Rightarrow \Delta^1+\Delta^2<0 \Rightarrow \Delta^1<0$ or $\Delta^2<0$. Using equalities (♠), the property follows. □

5.5.3 Insert inequalities are not $h_{\alpha,\beta}$-cuts

In the experimental results presented in Chapter 6, one can observe that neither insert inequalities nor swap inequalities do not improve the linear relaxation value of Formulation F^2. In the following property, we prove this results for insert inequalities. We try to adapt the proof to swap inequalities, but do not succeed to conclude.

Property 5.10

The insert inequalities are not $h_{\alpha,\beta}$-cut for Formulation F^2,
i.e. $\forall Y \in \underset{x \in P^n_{\delta,X}}{\arg\min}\, h_{\alpha,\beta}(\delta, X)$, $\forall u \in J$, *Y satisfies both (I_u) and (I'_u).*

Proof: Let us assume that there exist $Y^* \in \underset{(\delta,X)\in P^n_{F^2}}{\arg\min}\ h_{\alpha,\beta}(\delta, X)$ and $u_0 \in J$ such that Y^* does not satisfy (I_u).

By denoting $Y^* = (\delta, X)$, we then have $\Delta_{u_0}(\delta) < -M_{u_0}\,(1-\delta_{u_0})$.
By definition of M_u, one can deduce that $\delta_{u_0} \neq 0$, otherwise (I_u) would be valid.
Therefore, we can build a point $Y' = (\delta', X')$ that is different from Y^* as follows.

$$\forall i \in J,\ \delta'_j = \begin{cases} \delta_j \text{ if } j \neq u_0 \\ 0 \text{ otherwise} \end{cases} \quad \text{and} \quad \forall (i,j) \in J^<,\ X'_{i,j} = \begin{cases} X_{i,j} \text{ if } u_0 \notin \{i,j\} \\ \delta_j \text{ if } u_0 = i \\ \delta_i \text{ if } u_0 = j \end{cases}$$

Note that, if Y^* is an integer point, Y' is the points obtained by applying the insertion of u on the tardy side.

One can check that (δ', X') satisfies $(X.1 - X.4)$. Hence, $Y' \in P^n_{F^2}$, then we have $h_{\alpha,\beta}(\delta', X') \geqslant h_{\alpha,\beta}(\delta, X)$ by minimality of Y^*, that is $h_{\alpha,\beta}(\delta', X') - h_{\alpha,\beta}(\delta, X) \geqslant 0$. This difference can be written as follows.

$$\begin{aligned}
h_{\alpha,\beta}(\delta', X') - h_{\alpha,\beta}(\delta, X) = \quad & \alpha_{u_0} \sum_{j \in A(u_0)} \frac{p_j}{2} \Big[(\underbrace{\delta'_{u_0}}_{=0} + \underbrace{\delta'_j - X'_{u_0,j}}_{=0}) - (\delta_j + \delta_{u_0} - X_{u_0,j}) \Big] \\
& + \beta_{u_0} \sum_{j \in B(u_0)} \frac{p_j}{2} \Big[(\cancel{2} - \underbrace{\delta'_j}_{=\delta_j} - \underbrace{\delta'_{u_0}}_{=0} - \underbrace{X'_{u_0,j}}_{=\delta_j}) - (\cancel{2} - \delta_j - \delta_{u_0} - X_{u_0,j}) \Big] \\
& + \beta_{u_0} \Big[(\cancel{1} - \underbrace{\delta'_{u_0}}_{=0})\, p_{u_0} - (\cancel{1} - \delta_{u_0})\, p_{u_0} \Big] \\
& + \sum_{\substack{u \in J \\ u_0 \in A(u)}} \alpha_u \frac{p_{u_0}}{2} \Big[(\underbrace{\delta'_{u_0}}_{=0} + \underbrace{\delta'_j - X'_{u_0,j}}_{=0}) - (\delta_u + \delta_{u_0} - X_{u_0,u}) \Big] \\
& + \sum_{\substack{u \in J \\ u_0 \in B(u)}} \beta_u \frac{p_{u_0}}{2} \Big[(\cancel{2} - \underbrace{\delta'_u}_{=\delta_u} - \underbrace{\delta'_{u_0}}_{=0} - \underbrace{X'_{u_0,u}}_{=\delta_u}) - (\cancel{2} - \delta_u - \delta_{u_0} - X_{u_0,u}) \Big] \\
= \quad & \alpha_{u_0} \sum_{j \in A(u_0)} p_j \frac{-\delta_j - \delta_{u_0} + X_{u_0,j}}{2} && \text{(a)} \\
& + \beta_{u_0} \sum_{j \in B(u_0)} p_j \frac{-\delta_j + \delta_{u_0} + X_{u_0,j}}{2} && \text{(b)} \\
& + \beta_{u_0}\, \delta_{u_0}\, p_{u_0} && \text{(c)} \\
& + p_{u_0} \sum_{u \in \bar{A}(u_0)} \alpha_u \frac{-\delta_u - \delta_{u_0} + X_{u_0,u}}{2} && \text{(d)} \\
& + p_{u_0} \sum_{u \in \bar{B}(u_0)} \beta_u \frac{-\delta_u + \delta_{u_0} + X_{u_0,u}}{2} && \text{(e)}
\end{aligned}$$

Using the expressions of $\Delta_u(\delta)$ and M_u provided in pages 155 and 155 for $u=u_0$, we obtain the following equation.

$$\Delta_{u_0}(\delta) + M_{u_0}\,(1-\delta_{u_0}) = -\,\alpha_u \sum_{j\in A(u_0)} p_j\,(\delta_j + (1-\delta_{u_0})) \tag{A}$$

$$+\,\beta_u \sum_{j\in B(u)} p_j\,(1-\delta_j) + 0 \tag{B}$$

$$+\,\cancel{\beta_{u_0}\,p_{u_0}} - \beta_{u_0}\,p_{u_0}\,(\cancel{1}-\delta_{u_0}) \tag{C}$$

$$+\,p_{u_0} \sum_{j\in \bar{A}(u)} \alpha_j\,(\delta_j + (1-\delta_{u_0})) \tag{D}$$

$$+\,p_{u_0} \sum_{j\in \bar{B}(u)} \beta_j\,(1-\delta_{u_0}) + 0 \tag{E}$$

For any $j\in J\setminus\{u_0\}$, we have $X_{j,u_0} \leqslant 2-\delta_j-\delta_{u_0}$ from inequality $(X.4)$, then we have:

$\rightarrow$ $X_{j,u_0}+\delta_j+\delta_{u_0}-2\,\delta_j \leqslant 2-2\,\delta_j$, that is $(X_{j,u_0}-\delta_j+\delta_{u_0}) \leqslant 2\,(1-\delta_j)$, or again $\dfrac{-\delta_j+\delta_{u_0}+X_{j,u_0}}{2} \leqslant (1-\delta_j)$.
By summing up for any $j\in B(u)$ (resp. $j\in \bar{B}(u)$), we deduce that (a) $\leqslant$ (A) (resp. (b) $\leqslant$ (B)).

$\rightarrow$ $X_{j,u_0}-\delta_j-\delta_{u_0} \leqslant 2-2\,\delta_j-2\,\delta_{u_0}$, that is $\dfrac{-\delta_j-\delta_{u_0}+X_{j,u_0}}{2} \leqslant 1-\delta_j-\delta_{u_0}$.
By summing up for any $j\in A(u)$ (resp. $j\in \bar{A}(u)$), we deduce that (d) $\leqslant$ (D) (resp. (e) $\leqslant$ (E)).

Since (c)$=$(C), we finally have $h_{\alpha,\beta}(\delta', X') - h_{\alpha,\beta}(\delta, X) \leqslant \Delta_{u_0}(\delta) + M_{u_0}\,(1-\delta_{u_0}) < 0$. A contradiction.
Hence, a minimizer Y^* of $h_{\alpha,\beta}$ on $P^n_{F^2}$ necessarily satisfies (I_u).
Following the same lines, one can proof that it also satisfies (I'_u). □

In the next chapter, we provide experimental results to assess the practical efficiency of swap and insert inequalities for Formulation F^2.

Chapter 6

Practical application of dominance inequalities for UCDDP

In this chapter, we present how insert and swap inequalities can be used in combination with formulation F^2 to solve UCDDP. In this chapter, the expression "dominance inequalities" will only refer to insert and swap inequalities.

All experiments are carried out using a single thread with Intel(R) Xeon(R) X5677, @ 3.47GHz, and 144Gb RAM. MIP and LP are solved with CPLEX 12.6.3.0. The numerical experiments are performed on Biskup and Feldmann's benchmark [9], described on page 89. In this chapter, we use instances with $n \in \{10, 20, 50, 100, 200\}$. Moreover, to accurately compare formulations, we construct instances with $n \in \{60, 80\}$ (resp. $n \in \{120, 150, 180\}$) by only considering the first n tasks of the 100-task (resp. 200-task) instances provided in the benchmark. Unless otherwise specified, the gap, time and number of nodes presented in the following tables are average values over the ten instances for a given n and the time limit is set to 3600 seconds.

• *Formulations and formulation settings*

To measure the improvement induced by the insert or swap inequalities, we compare the four following formulations.

F^2: the formulation defined in Section 0.5, containing only inequalities $(X.1-X.4)$

F^i: the formulation obtained from F^2 by adding inequalities (I_u) and (I'_u) for all $u \in J$

F^s: the formulation obtained from F^2 by adding inequalities $(S_{u,v})$ for all $(u,v) \in J^2$ s.t. $u \neq v$

F^{i+s}: the formulation obtained from F^2 with additional inequalities of both F^i and F^s

For a given formulation F, we distinguish two settings: a setting with all available features, that is using CPLEX default, denoted by F_d, and a setting with less CPLEX features, denoted by F_l. Two types of features are disabled in this setting: the cut generation, which produces reinforcement inequalities and add them to the formulation; and the primal heuristic procedures. The cut generation is disabled in order to measure the impact of the dominance inequalities on the linear relaxation value of the formulation F^2, rather than their impact on the linear relaxation value of a strengthened formulation. The primal heuristic procedures have been disabled to focus on the lower bound since we have other methods to quickly obtain good feasible solutions (*Cf.* Section 6.3).

This results in eight formulation settings: $F_l^2, F_l^i, F_l^s, F_l^{i+s}, F_d^2, F_d^i, F_d^s$ and F_d^{i+s}. For each one, inequalities $(X.1-X.4)$, along with inequalities (I_u), (I'_u), and $(S_{u,v})$ when included, are added initially.

Let us recall that only the δ variables need to be integer in F^2. Indeed, from Lemma 0.5, if $\delta \in \{0,1\}^J$, then inequalities $(X.1-X.4)$ ensure that $X \in \{0,1\}^{J^<}$. It is also the case for F^i, F^s and

F^{i+s}. Therefore, unless otherwise specified, δ variables are set as binary variables, while X variables are set as continuous variables. Consequently, the branching decisions only involve δ.

> *NB:* Due to their large number of columns, Tables 6.1, 6.1, 6.3 and 6.4 have been placed at the end of the chapter, *i.e.* on pages 180 and 181.

6.1 Solving MIP formulations to optimality

Table 6.1 provides the results obtained when solving MIP to optimality, using the eight formulation settings. Each line corresponds to the ten instances of a given size n. More precisely, Table 6.1 entries are the following.

- #opt: the number of instances solved to optimality within the time limit
- time: the average running time in seconds over the instances solved to optimality
- #nd: the average number of nodes, except the root node, in the search tree, over the instances solved to optimality

For a given formulation setting, we choose to stop the run at a line of the table if less than five over ten instances are solved to optimality. For the subsequent lines, we report a "-" in the table.

Using formulation setting F_l^2, the ten n-task instances are solved to optimality within the time limit for n up to 50. In contrast, using F_l^i, it is the case for n up to 60, using F_l^s for n up to 120 and using F_l^{i+s} for n up to 150. Within approximately 5 minutes, F_l^2 solves 50-task instances, F_l^i 60-task instances, F_l^s 100-task instances and F_l^{i+s} 120-task instances. This computation time decrease is due to a drastic reduction in the number of nodes. For example, for $n=50$ the number of nodes goes from more than 53 000 for F_l^2 to only 31 for F_l^{i+s}. With this latter formulation setting, the number of nodes is low, it is at most 200, even for large size instances. However, the time limit is reached for some 180- and 200-task instances, since the size of the linear program solved at each node is large.

In the light of the first four columns of Table 6.1, we can conclude that, with less CPLEX features, adding insert and swap inequalities significantly reduces the number of nodes and hence the computation time. More precisely, adding only swap inequalities is better than adding only insert inequalities, but adding both of them provides the best performance.

The four last columns show the same improvement in terms of computation time and number of nodes when CPLEX features are used.

Let us now focus on the 4th and 5th columns to compare the impact of the dominance inequalities and the impact of all the CPLEX features. For small instances, *i.e.* $n \in \{10, 20\}$, Cplex features allow to solve the problem at the root node (*Cf.* F_d^2 columns). However, from $n=50$, the number of nodes grows fast, so that no 60-task instance can be solved within 3600 seconds. Conversely, we already noticed that adding swap and insert inequalities limits the number of nodes (*Cf.* F_l^{i+s} columns), so that the ten 150-task instances are solved within 3600 seconds. Finally, adding the swap and insert inequalities provides better results than adding CPLEX features.

For instances up to size 60, F_d^{i+s} solves the problem at the root node, and is faster than F_l^{i+s}. For larger instances, except 200-task instances, F_l^{i+s} and F_d^{i+s} solve the problem in similar computation times, even if F_l^{i+s} has a smaller number of nodes: for example, it is two times smaller for $n=150$. For $n=200$, F_d^{i+s} solves 4 over 10 instances, while F_l^{i+s} only solves 1 over 10 instances. To conclude, F_l^{i+s} and F_d^{i+s} offer comparable performances. The formulation providing the best results is F^{i+s} for both settings.

6.2 Lower bound obtained at the root node

To further investigate the impact of dominance inequalities, we focus in this section on the root node of the search tree for different formulation settings. More precisely, we compare the different lower bounds obtained at the root node.

In the CPLEX framework, setting the node limit to 0 allows to only solve the root node of a MIP: the Branch-and-Bound algorithm is stopped before the first branching. If CPLEX features are activated, the preprocessing is applied and the cuts are added before the algorithm stops. For a given formulation setting F, the corresponding run with the node limit set to 0 is denoted by F-RN. This results in eight runs: F_l^2-RN, F_l^i-RN, F_l^s-RN, F_l^{i+s}-RN, F_d^2-RN, F_d^i-RN, F_d^s-RN, F_d^{i+s}-RN.

Note that, in the CPLEX framework, solving F-RN is different from solving the linear relaxation of F, denoted by F-LP. Indeed, F-LP is obtained by setting δ variables as continuous variables, which desactivates most of CPLEX features. In particular, the reinforcement cuts cannot be added since they are not valid for the relaxed formulation. Similarly, the inference procedure on the binary variables cannot be applied. We run the four linear relaxations F^2-LP, F^i-LP, F^s-LP, and F^{i+s}-LP. We observed that the obtained values are the same for these four relaxations. In other words, adding insert and swap inequalities does not improve the linear relaxation value on this benchmark, which is consistent with Property 5.10, and which lets us think that swap inequalities are not $h_{\alpha,\beta}$-cut either. Since the values are the same, we only present in Table 6.2 the results for F^2-LP.

To measure the quality of the nine obtained lower bounds, we compute, when it is possible, the optimality gap, *i.e.* $(OPT - LB)/OPT$ where OPT denotes the optimal value and LB the lower bound. When the optimal value is not known, we compute a gap using UB the best upper bound that we get, *i.e.* $(UB - LB)/UB$. Such gaps are indicated with a "*" in Table 6.2. For each of the nine runs, the entries of Table 6.2 are the following.

L-gap: the average optimality gap of the lower bound obtained at the root node

time: the average running time in seconds over the ten instances

The obtained lower bound is exactly the same using either F^2-LP, F_l^2-RN, or F_l^s-RN. We deduce that with less CPLEX features and without insert inequalities, setting the δ variables as binary or continuous variables, provides the same lower bound. Moreover, this lower bound is quite weak, since the average optimality gap is larger than 40% even for the 10-task instances. The computation times using F^2-LP and F_l^2-RN are similar: 2 seconds for the 100-task instances and about 12 minutes for the 500-task instances. The computation time required for F_l^s-RN is larger: almost 20 seconds for the 100-task instances and 47 minutes for the 500-task instances.

The lower bound obtained when only considering the insert inequalities is slightly better when the δ variables are set as binary variables for $n \in \{10, 20\}$. Indeed, the average optimality gap is 33% instead of 41% when $n = 10$, and 66% instead of 68% when $n = 20$ (*Cf.* F^2-LP and F_l^i-RN columns). The computation time using F_l^i-RN is comparable to the computation time using F^2-LP and F_l^2-RN.

The lower bound obtained when considering both insert and swap inequalities, is significantly better when the δ variables are set as binary variables. Indeed, the average optimality gap is smaller than 39% for any value of n, and it is equal to 0 for $n = 10$ (*Cf.* F_l^{i+s}-RN column). The computation time for F_l^{i+s}-RN is between those for F_l^i-RN and F_l^s-RN: 14 seconds for the 100-task instances and about 30 minutes for the 500-task instances.

The lower bound provided using F_d^2-RN, is better than the one obtained using F_l^2-RN, that is with

less CPLEX features. Indeed, the average optimality gap is 7% instead of 41% when $n=10$, and 46% instead of 94% when $n=120$. However, the lower bound is weaker than the one obtained for F_l^{i+s}, whose optimality gap is 0 for $n=10$ and 38% for $n=120$. Moreover, the computation times using F_d^2-RN increase fast with the increase of n so that the root node cannot be solved within one hour for sizes larger than 120.

Combining CPLEX features with insert inequalities gives almost the same results (*Cf.* F_d^i-RN column). Conversely, combining CPLEX features with swap inequalities gives better results (*Cf.* F_d^s-RN column). In particular, the average computation time is reduced so that instances up to size 200 can be solved. Moreover, the optimality gap is less than 22% for all solved instances. Finally, using F_d^{i+s}-RN gives even better results, the average optimality gap does not exceed 15%, even for 200-task instances, which are solved in 418 seconds, instead of 1200 using F_d^s-RN.

In a nutshell, combining insert and swap inequalities is the best to obtain a good lower bound at the root node. Not using all the CPLEX features allows its fast computation (*Cf.* F_l^{i+s}-RN column). Conversely, using them allows to obtain a better lower bound at the expense of the computation time (*Cf.* F_d^{i+s}-RN column).

6.3 Using swap and insert inequalities to obtain an upper bound

In this section, we propose two upper bounds on the optimal value. The first one is derived from the fractional solution obtained at the root node by a simple rounding procedure. The second one is obtained by applying in addition a local search procedure.

• *A simple rounding procedure*

We derive an integer solution (δ, X) by rounding a fractional solution $(\tilde{\delta}, \tilde{X})$, as follows.

$$\forall j \in J,\ \delta_j = \begin{cases} 0 \text{ if } \tilde{\delta}_j < \frac{1}{2} \text{ or } \left(\tilde{\delta}_j = \frac{1}{2} \text{ and } \alpha_j < \beta_j\right) \\ 1 \text{ otherwise} \end{cases} \text{ and } \forall (i,j) \in J^<,\ X_{i,j} = \begin{cases} 1 \text{ if } \delta_i \neq \delta_j \\ 0 \text{ otherwise} \end{cases}$$

By construction, (δ, X) satisfies inequalities $(X.1$-$X.4)$ (*Cf.* Lemma 0.5). It is thus a solution of F^2, and $h_{\alpha,\beta}(\delta, X)$ is an upper bound of the optimal value. However, it is not necessarily a solution for F^i, F^s and F^{i+s} formulations, since (δ, X) does not necessarily satisfy the insert and swap inequalities. In order to transform such a solution into a solution satisfying the dominance inequalities, we can benefit from the fact that each dominance inequality is based on an operation as explained below.

• *Link between dominance properties and operations*

When a vector $\delta \in \{0,1\}^J$ encoding a partition (E,T) does not satisfy a dominance inequality, applying the corresponding operation to (E,T) provides a partition with a strictly lower penalty. The following property, formally states this result.

Property 6.1

Let $\delta \in \{0,1\}^J$ and let (E,T) be the partition encoded by δ.
For any $u \in J$,
(i) δ does not satisfy (I_u) for u ***if and only if*** *$u \in E$ and $f\Big(E \setminus \{u\}, T \cup \{u\}\Big) < f(E,T)$,*
(ii) δ does not satisfy (I'_u) for u ***if and only if*** *$u \in T$ and $f\Big(E \cup \{u\}, T \setminus \{u\}\Big) < f(E,T)$.*
Moreover, for any $(u,v) \in J^2$ such that $u \neq v$,
(iii) δ does not satisfy $(S_{u,v})$ for (u,v) ***iff*** *$(u,v) \in E \times T$ and $f\Big(E \setminus \{u\} \cup \{v\}, T \setminus \{v\} \cup \{u\}\Big) < f(E,T)$.*

Proof: Let us fix $u \in J$. From Property 5.5, δ does not satisfy inequality (I_u) if and only if (E,T) does not satisfy constraints $(\hat{I}_u)$, which is equivalent to $u \in E$ and $\Delta_u(E,T) < 0$. Using Property 5.4, it is equivalent to $u \in E$ and $f(E \setminus \{u\} \cup \{v\}, T \setminus \{v\} \cup \{u\}) - f(E,T) < 0$, which proves *(i)*. The proofs of *(ii)* and *(iii)* follow the same scheme. □

Corollary 6.2

Let $\delta \in \{0,1\}^J$ and let (E,T) be the partition encoded by δ.
(i) (E,T) is an insert local optimum ***if and only if*** *δ satisfies (I_u) and (I'_u) for all $u \in J$.*
(ii) (E,T) is a swap local optimum ***if and only if*** *δ satisfies $(S_{u,v})$ for all $(u,v) \in J^2$ s.t. $u \neq v$.*

• *A local search procedure based on insert and swap operations*

Let us call `Insert_swap_improvement` the procedure that consists, from a given initial solution, in iteratively apply the operation associated with each violated dominance inequality, until all of them are satisfied. Algorithm 1 presents a way to implement this procedure. From Property 6.1, if an

insert (resp. a swap) inequality is not satisfied, applying the appropriate insert (resp. swap) operation provides a strictly better solution. That ensures that each solution is considered at most one time in `Insert_swap_improvement` and thus that this procedure finishes. From Corrolary 6.2, the returned solution is an insert and swap local optimum, since it satisfies all the dominance inequalities.

`Insert_swap_improvement`

---

input: $\delta \in \{0,1\}^J$
output: δ' encoding an insert and swap local optimum

```
δ' ← δ; is_locally_opt ← false
while (not is_locally_opt)
  is_locally_opt ← true
  for u in J
    if δ' does not satisfy (I_u)
      δ'_u ← 0; is_locally_opt ← false
    if δ' does not satisfy (I'_u)
      δ'_u ← 1; is_locally_opt ← false
    for v in J\{u}
      if δ' does not satisfy (S_{u,v})
        δ'_u ← 0; δ'_v ← 1; is_locally_opt ← false
return δ'
```

Algorithm 1: The improvement procedure by insert and swap operations

Note that this algorithm can be seen as a local search procedure for the neighborhood associated to the insert and swap operations. Moreover, this procedure can be applied to any integer solution. Particularly, for the sake of comparison we apply it to the solutions obtained by heuristic "Heur II" proposed by Biskup and Feldmann in [9].

We finally compare the upper bounds given by the six following heuristic solutions.

BF : the solution obtained by Biskup and Feldmann's heuristic "Heur II"
$BF+$: the solution obtained by applying `Insert_swap_improvement` to BF
$R1$: the solution obtained by rounding the fractional solution of F^2-LP
$R1+$: the solution obtained by applying `Insert_swap_improvement` to $R1$
$R2$: the solution obtained by rounding the fractional solution of F_d^{i+s}-RN
$R2+$: the solution obtained by applying `Insert_swap_improvement` to $R2$

In the sequel, we will use the same notation for both a heuristic solution and its value, which provides an upper bound on the optimal value. To measure the quality of these upper bounds, Table 6.3 presents their optimality gap denoted by U-gap, *i.e.* $(UB - OPT)/OPT$ where OPT denotes the optimal value and UB the upper bound. The Biskup and Feldmann's heuristic provides a solution in less than 1 second. Applying rounding and `Insert_swap_improvement` to a fractional solution provides a solution in less than 1 second for instances up to size 200. Therefore, the time needed to obtain $R1$ and $R1+$ (resp. $R2$ and $R2+$) is essentially the computation time required to solve F^2-LP (resp. F_d^{i+s}-RN) already given in Table 6.2.

As shown in Table 6.3, BF is a good upper bound. Indeed, its optimality gap is smaller than 0.35% for instance sizes larger than 50. However, this bound is improved by `Insert_swap_improvement` : the optimality gap of $BF+$ is smaller than 0.02% for all the instances. With an optimality gap larger than 170%, $R1$ is a very weak upper bound, while $R1+$, with an optimality gap smaller than 0.01%,

is very good, and even slightly better than *BF*+. With an optimality gap smaller than 17%, *R*2 is a better upper bound than *R*1, and *R*2+ is exactly the same as *R*1+.

Finally, *BF*+, *R*1+ and *R*2+ are very good upper bounds. However, it is worth noticing that even if the computation time to obtain *BF*+ is about 1 second, the bound is obtained without any guarantee, since no lower bound is provided. Conversely, the computation time to obtain *R*2+ is larger: 25 seconds for $n=100$ and about 7 minutes for $n=200$, but a lower bound is provided. *R*2+ is then guaranteed to be at 14% of the optimal value for $n=100$, and at 15% for $n=200$ (*Cf.* L-gap of F_d^{i+s}-RN in Table 6.2). *R*1+ is a compromise between *BF*+ and *R*2+. Indeed, for instances up to size 200, *R*1+ is provided in less than 20 seconds together with a lower bound, but the guarantee obtained from this lower bound is quite weak (97% for $n=200$, *Cf.* F_l^2-RN in Table 6.2).

6.4 Insert and swap operations use cases

As shown in the previous sections, insert and swap operations can be used in different ways. Table 6.4 presents the best way to use them depending on the expected solution quality.

- To obtain an upper bound: apply rounding and `Insert_swap_improvement` to the fractional solution given by F^2-LP. (*Cf.* F^2-LP $\rightarrow R1+$ column in Table 6.4).
- To obtain an upper bound with a better guarantee than the one obtained with F^2-LP $\rightarrow R1+$: apply rounding and `Insert_swap_improvement` to the fractional solution given by F_d^{i+s}-RN. (*Cf.* F_d^{i+s}-RN $\rightarrow R2+$ column in Table 6.4).
- To obtain a 5%-approached solution: use F_d^{i+s}, setting the gap limit to 5%. (*Cf.* F_d^{i+s}-5% column in Table 6.4).
- To obtain an exact solution: use F_d^{i+s}. (*Cf.* F_d^{i+s} column in Table 6.4).

Table 6.4 sums up the performance of the four above mentioned use cases. To measure the performances on the 200-task instances, no time limit is fixed. The entries of Table 6.4 are the following.

L-gap: the average optimality gap of the provided lower bound
U-gap: the average optimality gap of the provided upper bound
time: the average running time in seconds
#nd: the average number of nodes except the root node

New experiments are conducted for the results reported in F_d^{i+s}-5% and F_d^{i+s} columns when $n=200$. These results are gathered with previously obtained results in Table 6.4 to offer a complete overview.

The limitation of the gap to 5% usually allows to save computation time by reducing the number of explored nodes. However, Table 6.4 shows that the computation time for F_d^{i+s}-5% is not much better than the computation time for F_d^{i+s}, only few seconds are saved. This is due to the small reduction of the number of nodes: the gap limit induces only a 2% reduction for $n=100$, and a 4% reduction for $n=200$.

The low impact of the 5% gap limit can be explained by two factors: the small number of nodes and the large size of the LPs solved at each node. Since insert and swap inequalities already reduces drastically the number of nodes, no space is left for further reduction. Moreover, for the six 200-task instances where F_d^{i+s} reaches the time limit, only less than 100 nodes are explored. The limit for solving F_l^{i+s} is thus the size or the difficulty of the LPs solved at each node, rather than the number of nodes.

Trying to address this issue, we have implemented a separation algorithm for the insert and swap inequalities using a callback function. The time needed to solve 50-task instances using this separation algorithm and CPLEX features was 1513 seconds with 925 nodes in average. We observed that 98% of the computation time was used by the UserCut Callback to add 71 inequalities in average. This is not surprising since the separation algorithm consists in simply evaluating the terms of inequality (I_u) and (I'_u) for the n possible tasks u, and the terms of inequality $(S_{u,v})$ for the n^2 possible couples (u, v), which results in an $O(n^3)$ procedure.

Providing a faster separation algorithm could reduce the computation time, but the branching scheme, and then the number of nodes, would be the same. Since this number of nodes is quite large compared to the performance of F_d^{i+s} (which solves all 50-task instances at the root node), we conclude that adding dominance inequalities through a separation procedure reduces their impact.

Indeed, when initially added, the dominance inequalities allow to CPLEX presolve phase to fix some variables to 0 or 1. The number of LPs variables is then reduced and the value obtained at each node is improved. When the δ variables are set as continuous variables, this presolve phase is not executed. It is then consistent with the observation that adding dominance inequalities in this latter case has no impact (*Cf.* Section 6.2).

	F_l^2			F_l^i			F_l^s			F_l^{i+s}			F_d^2			F_d^i			F_d^s			F_d^{i+s}		
n	#opt	time	#nd	#opt	time	#nd	#opt	time	#nd	#opt	time	#nd	#opt	time	#nd	#opt	time	#nd	#opt	time	#nd	#opt	time	#nd
10	10	29	11	10	34	10	10	32	7	10	0	0	10	26	0	10	22	0	10	3	0	10	0	0
20	10	51	162	10	63	91	10	63	25	10	42	11	10	44	0	10	54	0	10	41	0	10	10	0
50	10	311	53596	10	76	2101	10	90	56	10	67	31	10	1310	24725	10	156	1293	10	15	0	10	13	0
60	5	2078	228193	10	186	8063	10	74	83	10	58	41	0	-	-	5	439	2904	10	93	66	10	15	0
80	0	-	-	9	815	17604	10	137	138	10	77	70	-	-	-	2	2823	1402	10	219	322	10	79	73
100	-	-	-	4	2800	23965	10	291	215	10	109	75	-	-	-	-	-	-	10	529	542	10	165	141
120	-	-	-	-	-	-	10	728	269	10	219	122	-	-	-	-	-	-	10	1578	779	10	363	181
150	-	-	-	-	-	-	8	2532	410	10	786	201	-	-	-	-	-	-	2	3172	660	10	1011	481
180	-	-	-	-	-	-	1	3514	285	6	2460	194	-	-	-	-	-	-	-	-	-	5	1537	284
200	-	-	-	-	-	-	-	-	-	1	1929	127	-	-	-	-	-	-	-	-	-	4	2524	710

Table 6.1: Effect of adding insert and swap inequalities on exact solving

	F^2-LP		F_l^2-RN		F_l^i-RN		F_l^s-RN		F_l^{i+s}-RN		F_d^2-RN		F_d^i-RN		F_d^s-RN		F_d^{i+s}-RN	
n	L-gap	time	L-gap	time	L-gap	time	L-gap	time	L-gap	time	L-gap	time	L-gap	time	L-gap	time	L-gap	time
10	41%	0	41%	0	33%	0	41%	0	0%	0	7%	1	5%	1	0%	1	0%	0
20	68%	0	68%	0	66%	0	68%	0	12%	1	28%	2	28%	2	6%	1	2%	0
50	86%	0	86%	1	86%	1	86%	6	28%	6	42%	27	41%	31	17%	5	11%	3
60	89%	0	89%	1	89%	1	89%	7	36%	7	41%	91	41%	95	22%	9	16%	5
80	92%	1	92%	1	92%	1	92%	11	34%	8	43%	345	43%	359	21%	28	15%	10
100	93%	2	93%	2	93%	2	93%	19	35%	14	45%	1091	44%	1152	21%	62	14%	25
120	94%	3	94%	4	94%	11	94%	31	38%	15	46%	3189	46%	3192	22%	133	16%	52
150	96%	6	96%	13	96%	15	96%	60	34%	29	-	-	-	-	22%	352	15%	130
180	96%	12	96%	19	96%	23	96%	98	34%	49	-	-	-	-	22%	766	15%	274
200	97%	19	97%	25	97%	31	97%	126	39%	72	-	-	-	-	22%	1204	15%	418
500	99%*	722	99%*	698	99%*	742	99%*	2820	36%*	1870	-	-	-	-	-	-	-	-

Table 6.2: Effect of adding insert and swap inequalities on lower bounds

n	*BF* U-gap	*BF*+ U-gap	$R1$ U-gap	$R1$+ U-gap	$R2$ U-gap	$R2$+ U-gap
10	2.04%	0.00%	170%	0.00%	0.00%	0.00%
20	0.95%	0.00%	196%	0.00%	1.33%	0.00%
50	0.35%	0.02%	203%	0.00%	13.83%	0.00%
60	0.26%	0.01%	170%	0.01%	16.80%	0.01%
80	0.22%	0.01%	172%	0.00%	16.36%	0.00%
100	0.18%	0.00%	174%	0.00%	15.72%	0.00%
120	0.10%	0.00%	170%	0.00%	15.77%	0.00%
150	0.10%	0.00%	171%	0.00%	15.27%	0.00%
180	0.10%	0.00%	171%	0.00%	16.09%	0.00%
200	0.10%	0.01%	171%	0.01%	16.28%	0.00%

Table 6.3: Comparison of different heuristics providing an upper bound

to obtain:	an upper bound			a lower bound			a 5%-approximation		an exact solution	
use:	F^2-LP $\rightarrow R1$+			F_d^{i+s}-RN $\rightarrow R2$+			F_d^{i+s}-5%		F_d^{i+s}	
n	L-gap	U-gap	time	L-gap	U-gap	time	time	#nd	time	#nd
50	86%	0.00%	<1	11%	0.00%	3	8	0	4	34
100	93%	0.00%	2	14%	0.00%	25	160	114	165	141
200	97%	0.01%	20	15%	0.01%	418	7420	928	8317	1474
500	-	-(99%)	778	-	-	-	-	-	-	-

Table 6.4: Different ways of using insert and swap inequalities

Chapter 7

Dominance inequalities for other combinatorial optimization problems

In this Chapter, we propose some dominance inequalities for two classical combinatorial optimization problems: MAX-CUT and the maximum weigthed independent set. For both problems, we derive dominance inequalities from operations acting on solutions by using the general framework introduced in Chapter 5, and in particular Property 5.6.

For a given undirected graph $G=(V,E)$, let us denote by $N(u)$ the neighborhood of a node $u\in V$, *i.e.* $\boldsymbol{N(u)=\{v\in V\setminus\{u\}\mid\{u,v\}\in E\}}$.

7.1 Dominance inequalities for MAX-CUT

Let us present the compact MIP we use in this section to formulate MAX-CUT, which is defined in page 202. Let us consider an instance of the problem, that is $G=(V,E)$ an undirected graph, and $c\in\mathbb{R}^E$. A cut of G corresponds to a bi-partition of V, and then to two ordered bi-partitions of V, which can be encoded by n binary variables δ. We also use continuous variables X to linearize the objective function. In contrast with variables used in formulations F^2, F^3 and F^4, these variables are indexed by the edge set E, and not by $V^<$, which is not equivalent since G is not necessarily complete. We obtain the following formulation.

$$\boldsymbol{F^{\text{max-cut}}_{\delta,X}}: \max_{x\in \text{int}_\delta(P^G_{F^2})}(c\cdot X) \quad \text{where} \quad \boldsymbol{P^G_{F^2}}=\Big\{(\delta,X)\in\mathbb{R}^V\times\mathbb{R}^E \;\Big|\; (X.1-X.4)\Big\}$$

Note that this formulation is very similar to F^2. Actually, if G is complete, the solution sets are exactly the same. However, the objective functions have not the same structure. Therefore, even if the operations considered in this section are similar to insert and swap operations, the resulting dominance inequalities are different.

By analogy with the notation $S^n_{\delta,X}$, let us introduce the following set.

$$\boldsymbol{S^G_{\delta,X}}=\Big\{(\delta,X)\in\{0,1\}^V\times\{0,1\}^E \;\Big|\; (X.1-X.4)\Big\}$$

In the following pages we present dominance inequalities which can be used to reinforce Formulation $F^{max\text{-}cut}_{\delta,X}$. At this stage, we do not have neither experimental results to assess the practical efficiency of these inequalities, nor theoretical results stating whether these inequalities are c-cuts[1] or not. These questions are natural extensions of this work.

[1]For sake of brevity, we use c-cut instead of f_c-cut where f_c would be the objective function, *i.e.* $f_c=(\delta,X)\mapsto c\cdot X$ in this case.

7.1.1 Left-to-right insertion based dominance inequalities

• *Defining an operation on the solutions*

Let $u \in V$. We consider an operation equivalent to the insertion of a tardy task on the early side. Let us introduce $\mathscr{S}_u^+$ the set of vectors that encode an ordered bi-partition (E,T) such that $u \notin E$, *i.e.* $\mathscr{S}_{\boldsymbol{u}}^+ = \left\{(\delta,X) \in S_{\delta,X}^G \,\middle|\, \delta_u = 0\right\}$. Note that, if (E,T) is a partition encoded by a vector in $\mathscr{S}_u^+$, then removing u from T and adding it to E results in another partition. Let us introduce the corresponding operation θ_u^+ on the encoding vectors, using the function ψ^M defined in page 108 for $M = \{u\}$.

$$\theta_u^+ = \begin{pmatrix} \mathscr{S}_u^+ & \longrightarrow & S_{\delta,X}^G \\ (\delta,X) & \longmapsto & \psi^{\{u\}}(\delta,X) \end{pmatrix}$$

In order to apply Property 5.6, we have to identify suitable functions Δ and Π, before defining a constant M, and finally obtaining a dominance inequality.

• *Identifying function* Π

To identify points belonging to $\mathscr{S}_u^+$, we introduce the following linear function.

$$\Pi_u^+ = \begin{pmatrix} \mathbb{R}^V \times \mathbb{R}^E & \longrightarrow & \mathbb{R} \\ (\delta,X) & \longmapsto & \delta_u \end{pmatrix}$$

For any $(\delta,X) \in S_{\delta,X}^G$, we have $\Pi_u^+(\delta,X) \in \mathbb{N}$ and $\Pi_u^+(\delta,X) = 0 \Leftrightarrow \delta_u = 0 \Leftrightarrow (\delta,X) \in \mathscr{S}_u^+$.

• *Identifying function* Δ

The variation of the objective function between a vector (δ,X) in $\mathscr{S}_u^+$ and its image $\theta_u^+(\delta,X)$ is given by the following expression.

$$\Delta_u^+(\delta,X) = -\sum_{v \in N(u)} c_{\{u,v\}}\,\delta_v + \sum_{v \in N(u)} c_{\{u,v\}}\,(1-\delta_v) = \sum_{v \in N(u)} c_{\{u,v\}}\,(1-2\,\delta_v)$$

• *Defining constant* M

Since $\overset{\vee}{m}_u = \sum_{v \in N(u)} |c_{\{u,v\}}|$ is a positive upper bound of $\{\Delta_u^+(\delta,X) \,|\, (\delta,X) \in \mathscr{S}_u^+\}$, following the method described in page 157 for a maximization problem, we derive the following constant.

$$M_u^+ = \sum_{v \in N(u)} |c_{\{u,v\}}|$$

• *Finally writing a dominance inequality*

Following Property 5.6 adapted to a maximization problem, we obtain the inequality below, which translates the $\mathcal{N}^{\theta_u^+}$-dominance.

$$\underbrace{\sum_{v \in N(u)} c_{\{u,v\}}\,(1-2\,\delta_v)}_{\Delta_u^+(\delta,X)} \;\leqslant\; \underbrace{\Big(\sum_{v \in N(u)} |c_{\{u,v\}}|\Big)}_{M_u^+} \;\underbrace{\delta_u}_{\Pi_u^+(\delta,X)}$$

7.1.2 Right-to-left insertion based dominance inequalities

• *Defining an operation on the solutions*

Let $u \in V$. We consider an operation equivalent to the insertion of a early task on the tardy side. Let us introduce $\mathscr{S}_u^-$ the set of vectors that encode an ordered bi-partition (E,T) such that $u \in E$, *i.e.* $\mathscr{S}_{\boldsymbol{u}}^- = \left\{(\delta, X) \in S_{\delta,X}^G \,\middle|\, \delta_u = 1\right\}$. Note that, if (E,T) is a partition encoded by a vector in $\mathscr{S}_u^-$, then removing u from E and adding it to T results in another partition. Let us introduce the corresponding operation θ_u^- on the encoding vectors, using the function ψ^M defined in page 108 for $M = \{u\}$.

$$\theta_u^- = \begin{pmatrix} \mathscr{S}_u^- & \longrightarrow & S_{\delta,X}^G \\ (\delta, X) & \longmapsto & \psi^{\{u\}}(\delta, X) \end{pmatrix}$$

In order to apply Property 5.6, we have to identify suitable functions Δ and Π, before defining a constant M, and finally obtaining a dominance inequality.

• *Identifying function* Π

To identify points belonging to $\mathscr{S}_u^-$, we introduce the following linear function.

$$\Pi_u^- = \begin{pmatrix} \mathbb{R}^V \times \mathbb{R}^E & \longrightarrow & \mathbb{R} \\ (\delta, X) & \longmapsto & 1 - \delta_u \end{pmatrix}$$

For any $(\delta, X) \in S_{\delta,X}^G$, we have $\Pi_u^-(\delta, X) \in \mathbb{N}$ and $\Pi_u^-(\delta, X) = 0 \Leftrightarrow \delta_u = 1 \Leftrightarrow (\delta, X) \in \mathscr{S}_u^-$.

• *Identifying function* Δ

The variation of the objective function between a vector (δ, X) in $\mathscr{S}_u^+$ and its image $\theta_u^+(\delta, X)$ is given by the following expression.

$$\Delta_u^-(\delta, X) = + \sum_{v \in N(u)} c_{\{u,v\}}\, \delta_v - \sum_{v \in N(u)} c_{\{u,v\}}\, (1 - \delta_v) = \sum_{v \in N(u)} c_{\{u,v\}}\, (2\,\delta_v - 1)$$

• *Defining constant* M

Since M_u^+ is also a positive upper bound of $\{\Delta_u^-(\delta, X) \mid (\delta, X) \in \mathscr{S}_u^-\}$, let us set $M_u^- = M_u^+ = \sum_{v \in N(u)} |c_{\{u,v\}}|$.

• *Finally writing a dominance inequality*

Following Property 5.6 adapted to a maximization problem, we obtain the inequality below, which translates the $\mathcal{N}^{\theta_u^-}$-dominance.

$$\underbrace{\sum_{v \in N(u)} c_{\{u,v\}}\, (2\,\delta_v - 1)}_{\Delta_u^-(\delta, X)} \;\leqslant\; \underbrace{\Big(\sum_{v \in N(u)} |c_{\{u,v\}}|\Big)}_{M_u^-}\; \underbrace{(1 - \delta_u)}_{\Pi_u^-(\delta, X)}$$

7.1.3 Swapping based dominance inequalities

• *Defining an operation on the solutions*

Let[2] $(u,v) \in V^2$. We consider an operation equivalent to the swap between an early and a tardy task. The set of solutions where this operation can be applied, is the set $\mathscr{S}_{u,v}$ defined in page 160, where $S^n_{\delta,X}$ has to be changed into $S^G_{\delta,X}$. Moreover, the corresponding operation on the encoding vectors is given by $\theta_{u,v}$ defined on the same page. Finally, the function $\Pi_{u,v}$ allows to identify points belonging to $\mathscr{S}_{u,v}$.

Since the objective function of $F^{\text{max-cut}}_{\delta,X}$ is different from the one of F^2, function $\Delta_{u,v}$ and the constant $p(J)_{u,v}$ have to be redefined as follows.

• *Identifying function* Δ

The variation of the objective function between a vector (δ, X) in $\mathscr{S}_{u,v}$ and its image $\theta_{u,v}(\delta, X)$ is given by the following expression.

$$\Delta_{u,v}(\delta,X) = \sum_{w\in N(v)} c_{\{u,w\}}\,(1\text{–}2\,\delta_w) + \sum_{w\in N(u)} c_{\{v,w\}}\,(2\,\delta_w\text{–}1) + 2\,c_{\{u,v\}}\,\mathbb{I}_{u\in N(v)} \quad \text{where} \quad \mathbb{I}_{u\in N(v)} = \begin{cases} 1 \text{ if } u\in N(v) \\ 0 \text{ otherwise} \end{cases}$$

• *Defining constant* M

Note that $\mathbb{I}_{u\in N(v)}$ is a constant given by G. Moreover, $\overset{\vee}{m}_{u,v} = \sum_{w\in N(u)} |c_{\{u,w\}}| + \sum_{w\in N(v)} |c_{\{v,w\}}| + 2\,c_{\{u,v\}}\,\mathbb{I}_{u\in N(v)}$ is an upper bound of $\{\Delta_{u,v}(\delta,X) \,|\, (\delta,X)\in\mathscr{S}_{u,v}\}$.

Therefore, following the method proposed in page 157, we define the following constant.

$$M_{u,v} = \begin{cases} \overset{\vee}{m}_{u,v} & \text{if } \overset{\vee}{m}_{u,v} \geqslant 0 \\ \frac{\overset{\vee}{m}_{u,v}}{2} & \text{otherwise} \end{cases}$$

• *Finally writing a dominance inequality*

Following Property 5.6 adapted to a maximization problem, we obtain the inequality below, which translates the $\mathcal{N}^{\theta_{u,v}}$-dominance.

$$\sum_{w\in N(v)} c_{\{u,w\}}\,(1-2\,\delta_w) + \sum_{w\in N(u)} c_{\{v,w\}}\,(2\,\delta_w-1) + 2\,c_{\{u,v\}}\mathbb{I}_{u\in N(v)} \leqslant M_{u,v}\,(1-\delta_u+\delta_v)$$

We have not analyzed these inequalities in more detail, but it would be worth investigating further. In particular, an implementation of the proposed dominance inequalities would allow to test if they are able to reinforce Formulation $F^{max\text{-}cut}_{\delta,X}$.

[2]Note that an inequality is provided for each ordered pair of nodes, that is even for two nodes that are not adjacent in G, *i.e.* $\{u,v\}\notin E$.

7.2 Dominance inequalities for the maximum weighted independent set problem

Let us consider in this section an undirected graph $G=(V,E)$. A node subset $S\subseteq V$ is an **independent set**, or a stable set if no edge links two nodes in S, *i.e.* $\forall(u,v)\in S^2$, $\{u,v\}\notin E$. In other words, a node subset S is an independent set if each node in S has no neighbor in S. For a non-negative weight on the nodes, the maximum independent set problem aims at finding an independent set whose weight (the sum of the node weights) is maximum.

MAX. W. INDEP. SET		
	Input:	an undirected graph $G=(V,E)$ the node weights $c\in\mathbb{R}_+^V$
	Output:	an independent set $S\subseteq V$ maximizing $c(S)$

This problem is NP-hard since the unweighted version of the problem, *i.e.* when $c=\mathbb{I}_V$, is NP-hard [39].
One can note that, for any instance of MAX. W. INDEP. SET, the set of node subsets that do not include 0-weighted nodes is a dominant set. Therefore, the 0-weighted nodes can be removed from the instance. Without loss of generality, we can thus assume that the node weights are positive, *i.e.* $c\in(\mathbb{R}_+^*)^V$.

• *A linear formulation for* MAX. W. INDEP. SET
The node subsets, and then in particular the independent sets, can be encoded by binary variables $(x_v)_{v\in V}$. For $x\in\{0,1\}^V$, the encoded subset is $S=\{v\in V\mid x_v=1\}$ and its weights is $c\cdot x$. Moreover, S is an independent set if and only if x satisfies the following inequality.

$$\forall\{u,v\}\in E,\ x_u+x_v\leqslant 1 \tag{7.1}$$

This leads to the following classical compact IP formulation for MAX. W. INDEP. SET.

$$\boldsymbol{F}^{\text{edges}}:\ \max_{x\,\in\,P^{\text{edges}}\cap\mathbb{Z}^V} c\cdot x \quad\text{where}\quad \boldsymbol{P}^{\text{edges}}=\left\{x\in\mathbb{R}^V\,\middle|\,\forall v\in E,\ x_v\geqslant 0 \text{ and } x \text{ satisfies (7.1)}\right\}$$

We propose in this section, three dominance inequality families for this formulation.

7.2.1 Independent set adding based dominance inequalities

• *Defining an operation on the solutions*
Let us consider for this section $W\subseteq V$ an independent set. Let us introduce $\mathscr{S}_W$ the set of vectors that encodes an independent set that contains no nodes of W, *i.e.* $\mathscr{S}_{\boldsymbol{W}}=\left\{x\in S_x^G\,\middle|\,\forall w\in W,\ x_w=0\right\}$. Moreover, let us denote by $N(W)$ the set of neighbors of nodes in W that are not themselves in W, *i.e.* $\boldsymbol{N(W)}=\{v\in V\setminus W\mid\exists w\in W,\{w,v\}\in E\}$.
If S is an independent set encoded by a vector in $\mathscr{S}_W$, removing from S the neighbors of nodes of W and putting W in S results in another independent set. Let us introduce the corresponding operation θ_W on the encoding vectors.

$$\theta_W=\begin{pmatrix}\mathscr{S}_W & \longrightarrow & S_x^G\\ x & \longmapsto & x'\end{pmatrix}\quad\text{where}\quad\forall u\in V,\ x'_v=\begin{cases}1 & \text{if } v\in W\\ 0 & \text{if } v\in N(W)\\ x_v & \text{otherwise}\end{cases}$$

In order to apply Property 5.6, we have to identify suitable functions Δ and Π, before defining a constant M, and finally obtaining a dominance inequality.

• *Identifying function* Π

To identify points belonging to $\mathscr{S}_W$, we introduce the following linear function.

$$\Pi_W = \left(\begin{array}{ccc} \mathbb{R}^V & \longrightarrow & \mathbb{R} \\ x & \longmapsto & \sum\limits_{w \in W} x_w \end{array} \right)$$

For any $x \in S_x^G$, we have $\Pi_W(x) \in \mathbb{N}$ and $\Pi_W(x) = 0 \Leftrightarrow \forall w \in W,\ x_w = 0 \Leftrightarrow x \in \mathscr{S}_W$.

• *Identifying function* Δ

The variation of the objective function between a vector x in $\mathscr{S}_W$ and its image $\theta_W(x)$ is given by the following expression.

$$\Delta_W(x) = \sum_{w \in W} c_w \ - \sum_{u \in N(W)} c_u \, x_u$$

• *Defining constant* M

Since $\check{m}_W = c(W)$ is a positive upper bound of $\{\Delta_W(x) \mid x \in \mathscr{S}_W\}$, let us set $M_W = c(W)$.

• *Finally writing a dominance inequality*

Following Property 5.6 adapted to a maximization problem, we obtain the inequality below, which translates the $\mathcal{N}^{\theta_W}$-dominance.

$$\sum_{w \in W} c_w \ - \sum_{u \in N(W)} c_u \, x_u \leqslant c(W) \ \sum_{w \in W} x_w$$

This inequality, which can be reformulated as follows, is unfortunately not a c-cut (*Cf.* Property 7.1).

$$c(W) \left(1 - \sum_{w \in W} x_w \right) - \sum_{u \in N(W)} c_u \, x_u \leqslant 0 \qquad (I_W)$$

Property 7.1

Inequality (I_W) *is not a c-cut for Formulation* F^{edges}, *i.e.* $\forall x^* \in \underset{x \in P^{\text{edges}}}{\arg\min}(c \cdot x)$, x^* *satisfies* (I_W).

Proof: Let us assume that there exists $x^* \in \underset{x \in P^{\text{edges}}}{\arg\min}(c \cdot x)$ that violates (I_W).

We then have $c(W) \, (1 - \sum\limits_{w \in W} x^*_w) \ - \sum\limits_{u \in N(W)} c_u \, x^*_u > 0$ $(\bigstar)$.

By construction, one knows that $\Pi_W(x^*) \neq 0$, otherwise (I_W) would be valid.

Therefore, we can build a point x' that is different from x^* by setting $\forall u \in V$, $x'_v = \begin{cases} 1 & \text{if } v \in W \\ 0 & \text{if } v \in N(W) \\ x^*_v & \text{otherwise} \end{cases}$

One can check that x' satisfies (7.1). Indeed, for $\{u, v\} \in E$, one of the three following cases holds.

- If $\{u, v\} \cap W \neq \emptyset$, then only one node among u and v can belong to W, since W is an independent set. Without loss of generality, we assume that $u \in W$ and $v \in V \setminus W$. By definition, we then have $v \in N(W)$, therefore $x'_u + x'_v = 1 + 0 \leqslant 1$.
- If $\{u, v\} \cap N(W) \neq \emptyset$, we can then assume without loss of generality that $u \in N(W)$. Then we have $x'_u + x'_v = 0 + x'_v$ and by definition of x', $x'_v \leqslant 1$ since $x^*_v \leqslant 1$.
- Otherwise, $x'_u + x'_v = x^*_u + x^*_v \leqslant 1$.

Hence, $x' \in P^{\text{edges}}$, then we have $c \cdot x' \leqslant c \cdot x^*$ by maximality of x^*. However, the variation of the objective function between x^* and x' can be written as follows.

$$\begin{aligned} c \cdot x' - c \cdot x^* &= -\sum_{u \in N(W)} c_u \, x^*_u \; + \sum_{w \in W} c_w \, (1 - x^*_w) \\ &= -\sum_{u \in N(W)} c_u \, x^*_u \; + c(W) - \sum_{w \in W} c_w \, x^*_w \\ &\geqslant -\sum_{u \in N(W)} c_u \, x^*_u \; + c(W)(1 - \sum_{w \in W} x^*_w) \quad \text{since } \forall w \in W,\, c_w \leqslant c(W) \\ &> 0 \text{ by } (\bigstar) \end{aligned}$$

Therefore, $c \cdot x' > c \cdot x^*$. A contradiction. □

7.2.2 Edge swapping based dominance inequality

• *Defining an operation on the solutions*

Let us consider for this section $\{u, v\} \in E$ such that[3] $c_u < c_v$. By definition, $u \in N(v)$. Let us denote by $\widetilde{N}(v)$ the set of the neighbors of v except u, *i.e.* $\boldsymbol{\widetilde{N}(v) = N(v) \setminus \{u\}}$.

> *NB*: The notations $\mathscr{S}_{u,v}$, $\theta_{u,v}$, $\Pi_{u,v}$ and $\Delta_{u,v}$ introduced for the swap operation in Section 5.4 no longer apply, and these symbols will be used differently in this section.

Let us introduce $\mathscr{S}_{u,v}$ the set of vectors that encode an independent set that contains node u, necessarily not v, and not any other neighbor of v, *i.e.* $\boldsymbol{\mathscr{S}_{u,v}} = \left\{ x \in S^G_x \,\middle|\, x_u = 1,\, x_v = 0, \text{ and } \forall w \in \widetilde{N}(v), x_w = 0 \right\}$.
Note that the condition $x_v = 0$ is not useful, indeed, for $x \in S^G_x$, $x_u = 1 \Rightarrow x_v = 0$ since $\{u, v\} \in E$.
If S is an independent set encoded by a vector in $\mathscr{S}_{u,v}$, removing u from S and putting v instead results in another independent set. Let us introduce the corresponding operation $\theta_{u,v}$ on the encoding vectors.

$$\theta_{u,v} = \begin{pmatrix} \mathscr{S}_{u,v} & \longrightarrow & S^G_x \\ x & \longmapsto & x - \mathbb{I}_u + \mathbb{I}_v \end{pmatrix}$$

In order to apply Property 5.6, we have to identify suitable functions Δ and Π, before defining a constant M, and finally obtaining a dominance inequality.

• *Identifying function* Π

To identify points belonging to $\mathscr{S}_{u,v}$, we introduce the following linear function.

$$\Pi_{u,v} = \begin{pmatrix} \mathbb{R}^V & \longrightarrow & \mathbb{R} \\ x & \longmapsto & (1 - x_u) + \sum\limits_{w \in \widetilde{N}(v)} x_w \end{pmatrix}$$

For any $x \in S^G_x$, we have $\Pi_{u,v}(x) \in \mathbb{N}$ and $\Pi_{u,v}(x) = 0 \Leftrightarrow \begin{cases} x_u = 1 \\ \forall w \in \widetilde{N}(v),\, x_w = 0 \end{cases} \Leftrightarrow x \in \mathscr{S}_{u,v}$.

• *Identifying function* Δ

The variation of the objective function between a vector x in $\mathscr{S}_{u,v}$ and its image $\theta_{u,v}(x)$, is given by the following expression, which actually does not depend on x.

$$\Delta_{u,v}(x) = c_v - c_u$$

[3] Note that for some instances such edge does not exist, and in this case this section does not hold.

• *Defining constant M*

Since $\overset{\vee}{m}_{u,v} = c_v - c_u$ is a positive upper bound of $\{\Delta_{u,v}(x) \mid x \in \mathscr{S}_{u,v}\}$, let us set $M_{u,v} = c_v - c_u$.

• *Finally writing a dominance inequality*

Following Property 5.6 adapted to a maximization problem, we obtain the inequality below, which translates the $\mathcal{N}^{\theta_{u,v}}$-dominance.

$$c_v - c_u \leqslant (c_v - c_u)\left(1 - x_u + \sum_{w \in \widetilde{N}(v)} x_w\right)$$

This inequality, which can be simplified as follows (since $c_v - c_u > 0$), is not a c-cut (*Cf.* Property 7.2).

$$0 \leqslant -x_u + \sum_{w \in \widetilde{N}(v)} x_w \qquad (I_{u,v})$$

Property 7.2

Inequality $(I_{u,v})$ is not a c-cut for Formulation F^{edges}, i.e. $\forall x^* \in \arg\min_{x \in P^{edges}}(c \cdot x)$, x^* satisfies (I_W).

Proof: Let us assume that there exists $x^* \in \arg\min_{x \in P^{\text{edges}}}(c \cdot x)$ that violates $(I_{u,v})$.

We then have $x^*_u > \sum_{w \in \widetilde{N}(v)} x^*_w$ $(\bigstar)$.

By construction, one knows that $\Pi_{u,v}(x^*) \neq 0$, otherwise $(I_{u,v})$ would be valid.

Therefore, we can build a point x' that is different from x^* by setting $\forall w \in V$, $x'_w = \begin{cases} 1-m & \text{if } w = v \\ m & \text{if } w = u \\ x^*_w & \text{otherwise} \end{cases}$, where $m = \max_{w \in \widetilde{N}(v)} x^*_w$. By $(\bigstar)$, we have $m < x^*_u$.

One can check that x' satisfies (7.1). Indeed, for $\{a, b\} \in E$, one of the four following cases holds.

- If $\{a,b\} \cap \{u,v\} = \{u\}$, for example if $a = u$, then $x'_a = x'_u = m < x^*_u = x^*_a$, and $x'_b = x^*_b$ since necessarily $b \in V \setminus \{u, v\}$. Then $x'_a + x'_b < x^*_a + x^*_b \leqslant 1$ since x^* satisfies (7.1).
- If $\{a,b\} \cap \{u,v\} = \{v\}$, for example if $a = v$, then $x'_a = x'_v = 1 - m$, and $x'_b = x^*_b \leqslant m$ since necessarily $b \in V \setminus \{u, v\}$. Then $x'_a + x'_b \leqslant (1-m) + m \leqslant 1$.
- If $\{a,b\} \cap \{u,v\} = \{u,v\}$, then $x'_a + x'_b = x'_u + x'_v = (1-m) + m = m$.
- Otherwise, $\{a,b\} \cap \{u,v\} = \emptyset$, then $x'_a + x'_b = x^*_a + x^*_b \leqslant 1$ since x^* satisfies (7.1).

Hence, $x' \in P^{\text{edges}}$, then we have $c \cdot x' \leqslant c \cdot x^*$ by maximality of x^*. However, the variation of the objective function between x^* and x' can be written as follows.

$$\begin{aligned} c \cdot x' - c \cdot x^* &= -(c_u\, x^*_u + c_v\, x^*_v) + (c_u\, x'_u + c_v\, x'_v) \\ &= c_u\,(x'_u - x^*_u) + c_v\,(x'_v - x^*_v) \\ &= c_u\,(m - x^*_u) + c_v\,((1-m) - x^*_v) \\ &\geqslant c_u\,(m - x^*_u) + c_v\,((1-m) - (1-x^*_u)) \quad \text{since } x^*_v \leqslant 1 - x^*_u \text{ by (7.1)} \\ &= \underbrace{(c_u - c_v)}_{>0}\ \underbrace{(m - x^*_u)}_{>0} \\ &> 0 \end{aligned}$$

Therefore, $c \cdot x' > c \cdot x^*$. A contradiction. □

In the sequel, we propose a stronger version of inequality $(I_{u,v})$, by using the fact that $x_w = 0$ for any $w \in N(u)$ for a point x in $\mathscr{S}_{u,v}$, since $x_u = 1$, which allows to remove some terms of the sum $\sum_{w \in \widetilde{N}(v)} x_w$.

7.2.3 Stronger edge swapping based dominance inequality

We use in this section all the notations introduced for $\{u,v\} \in E$ such that $c_u < c_v$ in the previous section. In addition, we denote by $\widehat{N}(v)$ the set of the neighbors of v that are neither u, nor one of its neighbors, *i.e.* $\boldsymbol{\widehat{N}(v) = N(v) \setminus (\{u\} \sqcup N(u))}$.

One can remark that the conditions $x_w = 0$ for $w \in N(v) \cap N(u)$ are not useful in the definition of $\mathscr{S}_{u,v}$ since for $x \in S_x^G$, $x_u = 1 \Rightarrow \forall w \in N(u)$, $x_w = 0$. Therefore, we can use the following definition: $\mathscr{S}_{u,v} = \left\{ x \in S_x^G \;\middle|\; x_u = 1 \text{ and } \forall w \in \widehat{N}(v), x_w = 0 \right\}$. This definition leads to a new function $\widehat{\Pi}_{u,v}$ to identify points belonging to $\mathscr{S}_{u,v}$.

$$\widehat{\Pi}_{u,v} = \left(\begin{array}{ccc} \mathbb{R}^V & \longrightarrow & \mathbb{R} \\ x & \longmapsto & (1-x_u) + \sum\limits_{w \in \widehat{N}(v)} x_w \end{array} \right)$$

For any $x \in S_x^G$, we have $\widehat{\Pi}_{u,v}(x) \in \mathbb{N}$ and $\widehat{\Pi}_{u,v}(x) = 0 \Leftrightarrow x \in \mathscr{S}_{u,v}$. Using the function $\Delta_{u,v}$ and the constant $M_{u,v}$ defined in Section 7.2.2, we obtain the dominance inequality below according to Property 5.6.

$$0 \leqslant -x_u + \sum_{w \in \widehat{N}(v)} x_w \qquad (\widehat{I}_{u,v})$$

Note that the proof of Property 7.2 cannot be directly adapted for this new inequality. If the value of m is set to $\widehat{m} = \max_{w \in \widehat{N}(v)} x_w$, inequality (7.1) for $\{a,b\}$ where $a = v$ and $b \in N(v) \cap N(u)$ is not necessarily satisfied by x'. Indeed, we do not have $x'_b \leqslant \widehat{m}$ since $m \notin \widehat{N}(v)$. If the value of m is set to $\widetilde{m} = \max_{w \in \widetilde{N}(v)} x_w$ as previously, then we do not have $m < x^*_u$, since $(\bigstar)$ gives $\widehat{m} < x^*_u$.

Therefore, the question to know if inequality $\widehat{(I_{u,v})}$ is a c-cut for F^{edges} is open and should be investigated. On the other hand, we know that inequality $\widehat{(I_{u,v})}$ defines a facet of the polytope of the $\theta_{u,v}$-dominant solutions, which is defined as follows.

$$P^G_{u,v} = \text{conv}\left\{ x \in S_x^G \;\middle|\; x \in \mathscr{S}_{u,v} \Rightarrow c \cdot x \geqslant c \cdot \theta_{u,v}(x) \right\}$$

Property 7.3

Inequality $\widehat{(I_{u,v})}$ defines a facet of $P^G_{u,v}$.

Proof: Let us denote by $W = V \setminus (\widehat{N}(v) \sqcup \{u,v\})$. Let us consider the following points of $\mathbb{R}^V$ that belong to S_x^G since they encode independent sets.

- 0
- $\mathbb{I}_v$
- $\mathbb{I}_u + \mathbb{I}_w$ for any $w \in \widehat{N}(v)$
- $\mathbb{I}_w$ for any $w \in W$

One can check that these points satisfy to equality the inequality $\widehat{(I_{u,v})}$. In particular, this imply that these points are $\theta_{u,v}$-locally optimal since $\widehat{(I_{u,v})}$ translates the $\mathcal{N}^{\theta_{u,v}}$ dominance property.
Moreover, these $|V|$ points form an affinely independent family.
This shows that $\widehat{(I_{u,v})}$ defines a facet of $P^G_{u,v}$. □

A natural extension of this section would be an experimental works to test whether the proposed dominance inequalities are able to reinforce Formulation F^{edges}.

Conclusion

This section summarizes the main scientific contributions of the thesis and highlights some challenging issues.

In this thesis, we address two common due date problems, UCDDP and CDDP, through a polyhedral approach, with the aim to provide efficient linear formulations. As they are defined in their most general setting, that is without particular assumption on the unit earliness and tardiness penalties, the two considered problems are NP-hard.

• *Results for common due date problems*

Throughout the thesis, we extensively use the dominance properties to provide linear formulations for UCDDP and CDDP. More precisely, these properties enable to define a light encoding for schedules and to build linear inequalities: the so-called dominance inequalities. Most of the provided formulations are related to UCDDP since more dominance properties are available for UCDDP than for CDDP. To sum up, we provide:

- a formulation of UCDDP as a partitioning problem (*Cf.* F^1 in Section 0.5.1),
- a compact MIP formulation for UCDDP (*Cf.* F^2 in Section 0.5.2),
- a MIP formulation based on non-overlapping inequalities for UCDDP (*Cf.* F^3 in Section 1.3),
- a MIP formulation based on non-overlapping inequalities for CDDP (*Cf.* F^4 in Section 1.4),
- numerous facet defining inequalities that can be added to F^2, F^3, or F^4 (*Cf.* Chapter 4),
- three reinforcements of Formulation F^2 using dominance inequalities for UCDDP (*Cf.* F^i, F^s, and F^{i+s} defined in Chapter 6).

The proposed formulations (F^2, F^3, F^4, F^i, F^s, and F^{i+s}) have been implemented and their performances are compared to other methods on a standard benchmark. None of our formulations outperforms the state of the art method (Sourd [41]). However, our work on dominance properties shows that the efficiency of a formulation can be greatly improved when appropriate inequalities are added. Indeed, under a time limit of one hour, Formulation F^2 solves instances up to size 60, while F^{i+s}, where dominance inequalities have been added, solves instances up to size 200, and is notably better than the time-indexed formulation for some instances where processing times are especially long.

Beyond their numerical results, the proposed formulations are the result of a theoretical work that can be used in a larger context than the common due date scheduling field.

• *Contributions beyond the common due date problem*

In Part A, we provide some key lemmas about non-overlapping inequalities to use them in combination with other inequalities. We propose a method to provide pseudo-MIP formulations for scheduling problems using natural variables and non-overlapping inequalities. This method is applied on UCDDP (resp. CDDP) resulting in Formulation F^3 (resp. F^4), but could also be used for other scheduling problems as explained in Section 1.5.

The two last parts of the thesis share a common idea: reinforcing Formulation F^2 by adding inequalities that cut weak solutions. A first attempt is provided in Chapter 4, where the point 0 is cut by facet defining inequalities of the convex hull of the other solutions. The provided facets, which can be transposed as facets of the non-trivial cut polytope, are to numerous to be implemented, and do not seem to be promising for application purposes.

Nevertheless, Part B presents material that may be used beyond the common due date problems. Indeed, we propose in Chapter 3 a property that formally states how to transpose facet defining inequalities in general, and which enables us in particular to bridge the cut polytope, the boolean quadric polytope, and the polytope underlying F^2. Although widely used under different forms, this property can be interesting from a pedagogical point of view as it encompasses a wide range of results, like change of variables, lifting or symmetry results.

In Part C, we propose a new kind of inequality, called the dominance inequalities, to reinforce MIP formulations. These inequalities cut non-optimal integer solutions, while classical strengthening inequalities cut fractional points. We propose a simple method to derive a dominance inequality from an operation on the solutions, resulting in insert and swap inequalities for F^2.

Even if the experimental results show that insert and swap inequalities reduce the computation time, theoretical results show that they do not improve the linear relaxation value. Here, it would be interesting to further investigate in order to understand how insert and swap inequalities help the solver: are they able to improve the linear relaxation value in deeper nodes of the branching tree, when enough binary variables are set to 0 or 1? Are they taken into account only in preprocessing step that allow to fix a lot of binary variables before the Branch-and-Bound algorithm?

In Chapter 7, we attempt to define dominance inequalities for problems whose solutions have the same structure as a partition, like MAX-CUT and MAX. W. INDEP. SET. The dominance inequalities could be used for other combinatorial optimization problems, like the Traveling Salesperson Problem (TSP), at the expense of finding appropriate operations on solutions.

Bibliography

[1] *LEMON – Library for Efficient Modeling and Optimization in Networks.* (Cited on pages 85 and 89.)

[2] *PORTA – POlyhedron Representation Transformation Algorithm.* (Cited on page 118.)

[3] Emile Aarts and Jan Karel Lenstra, editors. *Local Search in Combinatorial Optimization.* Princeton University Press, 2003. (Cited on page 151.)

[4] Uttarayan Bagchi, Yih-Long Chang, and Robert S Sullivan. Minimizing absolute and squared deviations of completion times with different earliness and tardiness penalties and a common due date. *Naval Research Logistics*, 34(5):739–751, 1987. (Cited on pages 23 and 32.)

[5] Uttarayan Bagchi, Robert S. Sullivan, and Y. L. Chang. Minimizing mean absolute deviation of completion times about a common due date. *Naval Research Logistics Quarterly*, 33(2):227–240, 1986. (Cited on page 24.)

[6] Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988. (Cited on page 85.)

[7] Francisco Barahona and Ali Ridha Mahjoub. On the cut polytope. *Math. Program.*, 36(2):157–173, 1986. (Cited on pages 43 and 99.)

[8] Dirk Biskup and Martin Feldmann. Common due date scheduling. http://people.brunel.ac.uk/ mastjjb/jeb/orlib/schinfo.html. (Cited on page 89.)

[9] Dirk Biskup and Martin Feldmann. Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. *Computers & OR*, 28(8):787–801, 2001. (Cited on pages 29, 39, 89, 93, 171, and 176.)

[10] Peter Brucker. *Scheduling Algorithms.* Springer, 2006. (Cited on pages 9 and 10.)

[11] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer Programming.* Graduate Texts in Mathematics,Springer, 2014. (Cited on page 120.)

[12] Caterina De Simone. The cut polytope and the boolean quadric polytope. *Discrete Mathematics*, 79(1):71 – 75, 1990. (Cited on page 106.)

[13] A-E. Falq, P. Fouilhoux, and S. Kedad-Sidhoum. Mixed integer formulations using natural variables for single machine scheduling around a common due date. *Accepted in September 2020 for publication in Discrete Applied Mathematics*, abs/1901.06880, 2019. (Cited on page 20.)

[14] Lester Randolph Ford and Delbert R Fulkerson. A simple algorithm for finding maximal network flows and an application to the hitchcock problem. *Canadian journal of Mathematics*, 9:210–218, 1957. (Cited on page 202.)

[15] R. Fortet. L'algèbre de Boole et ses applications en recherche opérationelle. *Cahiers du Centre d'Études en Recherche Opérationnelle*, 4:5, 1959. (Cited on page 43.)

[16] Michael R. Garey, Robert E. Tarjan, and Gordon T. Wilfong. One-processor scheduling with symmetric earliness and tardiness penalties. *Math. Oper. Res.*, 13(2):330–348, 1988. (Cited on pages 25, 31, and 32.)

[17] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *ournal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961. (Cited on page 85.)

[18] Groupe GOThA. *Modèles et algorithmes en ordonnancement*. Ellipses, 2004. (Cited on page 9.)

[19] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981. (Cited on pages 37 and 87.)

[20] 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. (Cited on page 20.)

[21] 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. (Cited on pages 15, 25, 26, 31, and 32.)

[22] 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. (Cited on pages 20, 26, 31, and 32.)

[23] Antoine Jouglet and Jacques Carlier. Dominance rules in combinatorial optimization problems. *European Journal of Operational Research*, 212(3):433–444, 2011. (Cited on page 14.)

[24] Joanna Józefowska. *Just-In-Time Scheduling: Models and Algorithms for Computer and Manufacturing Systems*. International Series in Operations Research & Management Science. Springer, 2007. (Cited on page 11.)

[25] Helmut G. Kahlbacher. *Termin- und Ablaufplanung: ein analytischer Zugang*. PhD thesis, Kaiserslautern University of Technology, Germany, 1992. (Cited on page 32.)

[26] John J. Kanet. Minimizing the average deviation of job completion times about a common due date. *Naval research logistics quarterly*, Vol 28:643–651, Dec 1981. (Cited on pages 22 and 32.)

[27] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(1):373–395, 1984. (Cited on page 33.)

[28] Richard M. Karp. *Reducibility Among Combinatorial Problems*. The IBM Research Symposia Series. Plenum Press, New York, 1972. (Cited on page 202.)

[29] Franz-Josef Kramer and Chung-Yee Lee. Common due-window scheduling. *Production and Operations Management*, 2(4):262–275, 1993. (Cited on page 80.)

[30] Eugene L. Lawler. A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. (Cited on page 32.)

[31] Manfred Padberg. The boolean quadric polytope: Some characteristics, facets and relatives. *Math. Program.*, 45(1-3):139–172, 1989. (Cited on pages 93, 97, 99, 100, 103, 107, 108, 109, and 111.)

[32] Jean-Claude Picard and H. Donald Ratliff. Minimum cuts and related problems. *Networks*, 5(4):357–370, 1975. (Cited on pages 84 and 85.)

[33] Michael L. Pinedo. *Scheduling, Theory, Algorithms, and Systems*. Springer, 2016. (Cited on pages 9 and 10.)

[34] Maurice Queyranne. Structure of a simple scheduling polyhedron. *Mathematical Programming*, 58:263–285, 1993. (Cited on pages 3, 38, 49, 51, 53, and 83.)

[35] Maurice Queyranne and Andreas S. Schulz. Polyhedral approaches to machine scheduling. Technical Report 408, TU Berlin, 1994, revised 1996. (Cited on page 38.)

[36] M. Raghavachari. A v-shape property of optimal schedule of jobs about a common due date. *European Journal of Operational Research*, 23(3):401–402, 1986. (Cited on page 14.)

[37] R. Tyrell Rockafellar. *Convex Analysis*. Princetion University Press, 1970. (Cited on page 203.)

[38] Michel Sakarovitch. *Optimisation Combiantoire, tome 1: programmation dicretes*. Hermann, 1984. (Cited on page 35.)

[39] Alexander Schrijver. *Combinatorial optimization*. Algoritms and Combinatorics. Springer, 2002. (Cited on page 187.)

[40] Wayne E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66, 1956. (Cited on page 53.)

[41] Francis Sourd. New exact algorithms for one-machine earliness-tardiness scheduling. *INFORMS Journal on Computing*, 21(1):167–175, 2009. (Cited on pages 4, 30, 38, 89, and 193.)

[42] Shunji Tanaka, Shuji Fujikuma, and Mituhiko Araki. An exact algorithm for single-machine scheduling without machine idle time. *J. Sched.*, 12(6):575–593, 2009. (Cited on page 30.)

[43] Gérard Tisseau and Jacques Duma. *Tikz pour l'impatient*. 2017. (Cited on page 6.)

Appendices

Graph theory definitions

Definition GT.4

An (undirected)[4] ***graph*** *is a pair of sets* $G=(V,E)$*, where* $V\neq\emptyset$ *and* $E\subseteq\{\{u,v\}\mid(u,v)\in V^2, u\neq v\}$.
If $E=\{\{u,v\}\mid(u,v)\in V^2, u\neq v\}$*,* G *is the* ***complete*** *graph on* V.
The elements of V *are the* ***nodes*** *of* G*, while the elements of* E *are the* ***edges*** *of* G.

Definition GT.5

Let $G=(V,E)$ *be an undirected graph.*
$G'=(V',E')$ *is a* ***sub-graph*** *of* G *if* $V'\subseteq V$*,* $E'\subseteq E$ *and* G' *is a graph.*
The sub-graph of G ***induced*** *by* $V'\subseteq V$ *is* $G'=(V',E')$ *where* $E'=\{\{u,v\}\in E\mid u\in V', v\in V'\}$.

Definition GT.6

A ***clique*** *of an undirected graph* G *is a sub-graph of* G *that is complete.*

Definition GT.7

Let $G=(V,E)$ *an undirected graph. Let* $C\subseteq E$.
C *is an (elementary)*[5] ***path*** *of* G *if* | *there exists a sequence of* $l+1$ *distinct nodes* $(u_i)_{i\in[0..l]}\in V^{l+1}$ *such that* $C=\{\{u_i,u_{i+1}\}\mid i\in[0..l-1]\}$.
In this case:
- l *is called the* ***length*** *of* C
- u_0 *and* u_l *are called the* ***endpoints*** *of* C
- *the node subset* $\{u_i\mid i\in[0..l]\}$ *is called the* ***support*** *of* C*, and denoted by* $\mathbf{supp}(C)$.
- C *is an* ***Hamiltonian path*** *if* $\mathrm{supp}(C)=V$.
- *a node* $v\in V$ *is said* ***on*** C *if* $v\in\mathrm{supp}(C)$.
- *two nodes* v^1 *and* v^2 *on* C *are said* ***consecutive on*** C *if* $\{v^1,v^2\}\in C$.

Definition GT.8

Let $G=(V,E)$ *an undirected graph. Let* $C\subseteq E$.
C *is an (elementary)* ***cycle*** *of* G *if* | *there exists a sequence of* $l\geqslant 3$ *distinct nodes* $(u_i)_{i\in[1..l]}\in V^l$ *such that* $C=\Big\{\{u_i,u_{i+1}\}\mid i\in[1..l-1]\Big\}\cup\Big\{\{u_l,u_1\}\Big\}$.
In this case:
- l *is called the* ***length*** *of* C
- *the node subset* $\{u_i\mid i\in[0..l]\}$ *is called the* ***support*** *of* C*, and denoted by* $\mathbf{supp}(C)$.
- C *is an* ***Hamiltonian cycle*** *if* $\mathrm{supp}(C)=V$.
- *a node* $v\in V$ *is said* ***on*** C *if* $v\in\mathrm{supp}(C)$.
- *two nodes* v^1 *and* v^2 *on* C *are said* ***consecutive on*** C *if* $\{v^1,v^2\}\in C$.

[4] Since only *undirected* graphs will be considered in this thesis, we will only say "graph".
[5] Since only *elementary* paths (resp. cycles) will be considered in this thesis, we will only say "path" (resp. "cycle").

Definition GT.9

Let $G=(V,E)$ an undirected graph. Let $C \subseteq E$.
*C is a **cut** if there exists $W \subseteq V$ such that $C=\{\{u,v\} \in E \mid u \in W, v \in V \setminus W\}$.*
*In this case, C is called the **cut induced** by the partition $\{W, V \setminus W\}$.*

Let us define two classical optimization problems related to the cuts in a graph: MIN-CUT, and its generalization MAX-CUT.

MIN-CUT	Input:	an undirected graph $G=(V,E)$ non-negative weights on the edges $c \in (\mathbb{R}_+)^E$
	Output:	a subset of nodes $S \subseteq V$ minimizing $c(S : V \setminus S)$

MAX-CUT	Input:	an undirected graph $G=(V,E)$ weights on the edges $c \in \mathbb{R}^E$
	Output:	a subset of nodes $S \subseteq V$ maximizing $c(S : V \setminus S)$

Note that in spite of their great similarity, these problems belongs to different[6] complexity classes. Indeed, MIN-CUT $\in P$ since it can be solved by the Ford-Fulkerson algorithm [14], while MAX-CUT is NP-complete, as shown by Karp in [28].

[6] Unless $P=NP$

Convex analysis definitions and properties

CA.3 General properties and definitions

We recall here some well-known definitions and theorems in convex analysis. They can be found in the reference book of Rockafellar titled "*Convex Analysis*" [37]. Results are given in $\mathbb{R}^d$ (for $d \in \mathbb{N}^*$), even if most of them are also true for an arbitrary real vector space. We denote by $x \cdot y$ the scalar product between two vectors of $\mathbb{R}^d$, *i.e.* $x \cdot y = \sum_{i=1}^{d} x_i\, y_i$. Note that in finite-dimensional space, and thus in particular in $\mathbb{R}^d$, compact sets are bounded closed sets.

We give the definition of the different structures we use (affine space, convex, cone, polyhedron), some of their basis properties, along with some important theorems (Minkowski, Klee, Minkowski-Weyl), which ensure that such structures can be generated from a subset of key elements, as a vector space is generated by a basis.

In the sequel, $\mathbb{R}^d$, which is both a vector space and an affine space on itself, has to be seen sometimes as a vector set, sometimes as a point set. Moreover, several definitions are available for most of the considered structure. We chose one of them as "the definition", and propose equivalent definitions as remarks.

Let us start by defining some usual notations.

Definition CA.1

Let $(x, y) \in \mathbb{R}^d \times \mathbb{R}^d$.
The ***line*** *through* x *and* y, *is* $\{x + \lambda(y-x) \,|\, \lambda \in \mathbb{R}\} = \{\lambda y + (1-\lambda)x \,|\, \lambda \in \mathbb{R}\}$.
The ***line segment*** *between* x *and* y *is* $\{x + \theta(y-x) \,|\, \theta \in [0,1]\} = \{\theta y + (1-\theta)x \,|\, \theta \in [0,1]\}$.
The line segment between x *and* y *is denoted by* $[\boldsymbol{x}, \boldsymbol{y}]$ *or* $[\boldsymbol{y}, \boldsymbol{x}]$.

Definition CA.2

Let $A \subseteq \mathbb{R}^d$ *and* $B \subseteq \mathbb{R}^d$. *The* ***Minkowski sum*** *of* A *and* B *is* $\boldsymbol{A+B} = \{a+b \,|\, a \in A,\, b \in B\}$.

Remark CA.3

Note that if $A = \emptyset$ or $B = \emptyset$, then $A+B = \emptyset$.
Moreover, if $B = \{x\}$, then $A+B$ can be denoted by $A+x$ instead of $A+\{x\}$.

CA.3.1 Affine sets

Definition CA.4

Let $m \in \mathbb{N}^*$. Let $(x_i)_{i\in[1..m]} \in (\mathbb{R}^d)^m$. Let $y \in \mathbb{R}^d$.
*y is an **affine combination** of points $(x_i)_{i\in[1..m]} \Leftrightarrow \exists\, (\lambda_i)_{i\in[1..m]} \in \mathbb{R}^m$, $\sum_{i=1}^{m} \lambda_i = 1$ and $\sum_{i=1}^{m} \lambda_i x_i = y$*

Definition CA.5

Let $A \subseteq \mathbb{R}^d$.
*A is an **affine set** in $\mathbb{R}^d \Leftrightarrow \left.\begin{array}{l} \exists\, F \text{ a vectorial subspace of } \mathbb{R}^d \\ \exists\, x \in \mathbb{R}^d \end{array}\right\}$ such that $A = F + x$.*

*In this case, F is called the **direction** of A, and is denoted by $\overrightarrow{A}$.*
*The **dimension** of A is defined as the dimension of F, and denoted by $\mathbf{dim}(A)$.*

Remark CA.6

An affine set can be defined as a set containing the line through each pair of points that it contains, or as a set that is stable by affine combination, *i.e.* a set containing all the affine combinations of its points.

Remark CA.7

Note that $\overrightarrow{A} = A - A = \{a' - a \,|\, a \in A,\, a' \in A\}$

Property CA.8

Let A^1 and A^2 two affine sets of $\mathbb{R}^d$. $A^1 \cap A^2$ is either the empty set, or an affine set.
***If** $A^1 \cap A^2 = \emptyset$, **then** $\overrightarrow{A^1} \subseteq \overrightarrow{A^2}$ or $\overrightarrow{A^2} \subseteq \overrightarrow{A^1}$, **otherwise** $\overrightarrow{A^1 \cap A^2} = \overrightarrow{A^1} \cap \overrightarrow{A^2}$.*

Corollary CA.9

Let $S \subseteq \mathbb{R}^d$.
*There exists a smallest affine set containing S, called the **affine set generated** by S.*
This set is denoted by $\mathbf{aff}(S)$, and is given by $\mathrm{aff}(S) = \bigcap_{\substack{A \text{ affine} \\ S\subseteq A}} A$.

Definition CA.10

Let $S \subseteq \mathbb{R}^d$.
*The **dimension** of S is the one of the affine set that it generates, i.e. $\mathbf{dim}(S) = \dim(\mathrm{aff}(S))$.*

Definition CA.11

Let $m \in \mathbb{N}^*$. Let $(u_i)_{i\in[1..m]} \in \mathbb{R}^{d^m}$. Let $(x_i)_{i\in[0..m]} \in \mathbb{R}^{d^{m+1}}$.
*Vectors $(u_i)_{i\in[1..m]}$ are **linearly independent** if $\forall (\lambda_i)_{i\in[1..m]} \in \mathbb{R}^m$, $\sum_{i=1}^{m} \lambda_i u_i = 0 \Rightarrow \lambda = 0$.*
*Points $(x_i)_{i\in[0..m]}$ are **affinely independent** if $(x_i - x_0)_{i\in[1..m]}$ are linearly independent.*

Remark CA.12

In a linearly independent vector family, no vector can be written as a linear combination of the others. In other words, each vector does not belong to the vector space generated by the other vectors.

In an affinely independent point family, no point can be written as an affine combination of the others. In other words, each point does not belong to the affine set generated by the other points.

Remark CA.13

If $(u_i)_{i\in[1..m]}$ are linearly independent, then these m vectors generate a m-dimensionnal vectorial space.

If $(x_i)_{i\in[0..m]}$ are affinely independent, then these $m+1$ points generate a m-dimensionnal affine space.

CA.3.2 Convex sets

Definition CA.14

Let $m \in \mathbb{N}^*$. Let $(x_i)_{i\in[1..m]} \in (\mathbb{R}^d)^m$. Let $y \in \mathbb{R}^d$.
*y is a **convex combination** of points $(x_i)_{i\in[1..m]} \Leftrightarrow \exists\, (\lambda_i)_{i\in[1..m]} \in [0,1]^m$, $\sum_{i=1}^{m} \lambda_i = 1$ and $\sum_{i=1}^{m} \lambda_i\, x_i = y$*

Remark CA.15

y is the **barycenter** of weighted points $(x_i, \lambda_i)_{i\in[1..m]}$, *i.e.* y is the only point satisfying $\sum_{i=1}^{m} \lambda_i (y - x_i) = 0$.
However, barycenter definition is broader : weights $(\lambda_i)_{i\in[1..m]}$ can be arbitrary, provided that $\sum_{i=1}^{m} \lambda_i \neq 0$.

Definition CA.16

*Let $S \subseteq \mathbb{R}^d$. S is a **convex set** if and only if $\forall (x,y) \in S^2$, $\forall \theta \in [0,1]$, $\theta\, x + (1-\theta)\, y \in S$.*

Remark CA.17

A convex set can be defined as a set containing the segment between each pair of points that it contains, or as a set that is stable by convex combination.

Property CA.18

Let $(C^i)_{i\in\mathcal{I}}$ be a family of convex sets of $\mathbb{R}^d$, indexed by an arbitrary set $\mathcal{I}$.
The intersection $\bigcap_{i\in\mathcal{I}} C^i$ is also a convex set.
***If** $\mathcal{I}$ is a finite non-empty set, **then** the Cartesian product $\prod_{i\in\mathcal{I}} C^i$ is also a convex set.*
***If** $\mathcal{I}$ is a finite non-empty set, **then** the Minkowski sum $\sum_{i\in\mathcal{I}} C^i$ is also a convex set.*

Corollary CA.19

Let $S \subseteq \mathbb{R}^d$.
*There exists a smallest convex set containing S, called the **convex hull** of S.*
This set is denoted by $\mathbf{conv}(S)$, and is given by $\mathrm{conv}(S) = \bigcap_{\substack{C \text{convex} \\ S\subseteq C}} C$.

Remark CA.20

One can also define the convex hull of S as the set of convex combinations of points in S,
i.e. $\mathrm{conv}(S) = \left\{ \sum_{i\in I} \lambda_i\, s_i \;\middle|\; I \text{ finite set}, (s_i)_{i\in I} \in S^I,\ (\lambda_i)_{i\in I} \in [0,1]^I,\ \sum_{i\in I} \lambda_i = 1 \right\}$.

Theorem CA.21 (*Carathéodory*)

It suffices to consider convex combinations of at most $d+1$ points to have all the convex combinations, i.e. $\mathrm{conv}(S) = \left\{ \sum_{i=0}^{d} \lambda_i\, s_i \;\middle|\; (s_i)_{i\in[0..d]} \in S^{d+1},\ (\lambda_i)_{i\in[0..d]} \in [0,1]^{d+1},\ \sum_{i=0}^{d} \lambda_i = 1 \right\}$.

CA.3.3 Cones

Definition CA.22

*Let $S \subseteq \mathbb{R}^d$. S is a **cone** if and only if $\forall x \in A$, $\forall \lambda \in \mathbb{R}_+^*$, $\lambda\, x \in A$.*

Remark CA.23

Equivalently, a cone can be defined as a set that is stable by positive multiplication.
Similarly, a convex cone is as a set that is stable by positive linear combination.

Property CA.24

Let $\mathcal{I}$ be an arbitrary set.
***If** $(K^i)_{i\in\mathcal{I}}$ is a family of cones of $\mathbb{R}^d$, **then** $\bigcap_{i\in\mathcal{I}} C^i$ is also a cone.*
***If** $(K^i)_{i\in\mathcal{I}}$ is a family of convex cones of $\mathbb{R}^d$, **then** $\bigcap_{i\in\mathcal{I}} C^i$ is also a convex cone.*

Corollary CA.25

Let $S \subseteq \mathbb{R}^d$.
*There exists a smallest convex cone containing S, called the **cone generated** by S.*
This set is denoted by $\mathbf{cone}(S)$, and is given by $\bigcap_{\substack{K \text{ cvx cone} \\ S\subseteq K}} K$.

Let $\mathbf{cone^0}(S)$ denote the smallest convex cone containing $S \cup \{0\}$, i.e. $\text{cone}^0(S) = \text{cone}(S) \cup \{0\}$.

Remark CA.26

Beware, the notation does not show that $\text{cone}(A)$ has to be a convex set.
If K is a cone, **then** $\text{cone}\,(K) = \text{conv}(K)$.
If C is a convex set, **then** $\text{cone}\,(C) = \{\lambda\, c \,|\, \lambda \in \mathbb{R}_+^*, c \in C\}$
If F is a vector space, **then** $\text{cone}\,(F) = F$.

CA.3.4 Recession Cones

Definition CA.27

Let C be a convex set of $\mathbb{R}^d$. Let $u \in \mathbb{R}^d$.
*C **recedes** in the direction u if and only if $\forall c \in C$, $\forall \lambda \in \mathbb{R}_+$, $c + \lambda\, u \in C$.*
*One also say that u is a **recession direction** of C.*

Remark CA.28

Let C be a convex set of $\mathbb{R}^d$.
(i) 0 is a recession direction for C.
(ii) **If** u and v are both recession directions for C,
then $\forall (\lambda, \mu) \in {\mathbb{R}_+}^2$, $\lambda\, u + \mu\, v$ is also a recession direction for C.
Consequently, the set of recession directions for C is a convex cone containing 0.
This justifies the following definition.

Definition CA.29

Let C be a convex set of $\mathbb{R}^d$.
*The **recession cone** of C, denoted by $\mathbf{0^+(C)}$, is the set of recession directions for C.*

CA.3.5 Extreme points and extreme directions

Definition CA.30

Let C be a convex set of $\mathbb{R}^d$. Let $x \in C$.

*x is an **extreme point** of C* $\Leftrightarrow \forall (y,z) \in C \times C,\ \forall \theta \in]0,1[,\ x = \theta\, y + (1-\theta)\, z \ \Rightarrow\ x = y = z$

$\Leftrightarrow \forall (y,z) \in \mathbb{R}^d \times \mathbb{R}^d,\ x = \dfrac{y+z}{2} \ \Rightarrow\ x = y = z$

$\Leftrightarrow$ *x cannot be written as the middle of two other points in C*

The set of the extreme points of C is denoted by $\mathbf{extr}(C)$.

Remark CA.31

Note that $x \in \text{extr}(C)$ if and only if $C \setminus \{x\}$ is also a convex set.
Therefore, if $C = \text{conv}(S)$ then $\text{extr}(C) \subseteq S$, but in general $\text{extr}(C) \neq S$.
For example if S contains three aligned points, one of them cannot be extreme in $\text{conv}(S)$.

Lemma CA.32

Let C^1 and C^2 be two convex sets of $\mathbb{R}^d$.
***If** $C^1 \subseteq C^2$, **then** $\text{extr}\,(C^2) \cap C^1 \subseteq \text{extr}\,(C^1)$.*

Proof: If $x \in C^1 \cap \text{extr}(C^2)$, then x is a point of C^1 that cannot be written as convex combination of points in C^2. Hence, it cannot be either written as a convex combination of points in C^1. Thus $x \in \text{extr}\,(C^1)$. □

Remark CA.33

Note that in general the converse inclusion is not true, and hence no equality stands.
For a counter-example, let us consider $C^2 = \text{conv}\,\{a,b,c\}$ where a, b, c are three points not aligned and $C^1 = \text{conv}\,\{a',b',c'\}$, where $a' = \frac{a+b}{2}$, $b' = \frac{b+c}{2}$, $c' = \frac{a+c}{2}$. In this case, $\text{extr}\,(C^2) \cap C^1 = \{a,b,c\} \cap C^1 = \emptyset$, while $\text{extr}\,(C^1) = \{a',b',c'\} \neq \emptyset$.
However, in the case where C^2 is convex, and C^1 one of its faces, there is equality (*Cf.* Lemma CA.56).

Lemma CA.34

Let $m \in \mathbb{N}^*$. Let $(C^i)_{i \in [1..m]}$ be a family of m convex sets of $\mathbb{R}^d$.
If $\prod$ denotes the cartesian product, we have $\text{extr}\left(\prod_{i=1}^{m} C^i\right) = \prod_{i=1}^{m} \text{extr}(C^i)$.

Remark CA.35

If K is a convex cone, **then** $\text{extr}(K) = \{0\}$ or $\text{extr}(K) = \emptyset$.
In other words, extreme points are not the good notion for cones, which have to be considered more as sets of directions than as sets of points.

Definition CA.36

Let K be a convex cone of $\mathbb{R}^d$. Let $u \in K$ such that $u \neq 0$.

*u is an **extreme direction** of K* $\Leftrightarrow \left.\begin{array}{c} \forall (v,w) \in K \times K \\ \forall (\lambda,\mu) \in \mathbb{R}_+^* \times \mathbb{R}_+^* \end{array}\right\},\ u = \lambda\, v + \mu\, w \Rightarrow v \in \mathbb{R}_+\, u, w \in \mathbb{R}_+ u$

$\Leftrightarrow$ *u cannot be written as the positive combination of two other directions in K*

The set of the extreme directions of K is denoted by $\overrightarrow{extr}(K)$.

CA.3.6 Generation of a convex set by its extreme points and directions

Theorem CA.37 (*Minkowski*)

Let $C \subseteq \mathbb{R}^d$ *such that* $C \neq \emptyset$. ***If*** $\begin{cases} C \text{ is a convex set} \\ C \text{ is compact} \end{cases}$ ***then*** $C = \text{conv}(\text{extr}(C))$.

Corollary CA.38

Let C *be a compact convex set of* $\mathbb{R}^d$.
(i) $C \neq \emptyset \Leftrightarrow \text{extr}(C) \neq \emptyset$
(ii) $\dim(C) = \dim(\text{extr}(C))$

Proof: • Since $\text{extr}(C) \subseteq C$, $C = \emptyset \Rightarrow \text{extr}(C) = \emptyset$.
Conversely, if $C \neq \emptyset$, then we have $C = \text{conv}(\text{extr}(C))$ by Theorem CA.37.
Therefore, $\text{extr}(C) = \emptyset$ would imply that $C = \text{conv}(\emptyset) = \emptyset$, which is not true. Thus $\text{extr}(C) \neq \emptyset$.

• In general, for a set $S \subseteq \mathbb{R}^d$, we have $\dim(\text{conv}(S)) = \dim(\text{aff}(\text{conv}(S))) = \dim(\text{aff}(S)) = \dim(S)$, since an affine combination of convex combinations of points is an affine combination of these points.
In particular, since $C = \text{conv}(\text{extr}(C))$ according to Theorem CA.37, we get $\dim(C) = \dim(\text{extr}(C))$. □

Property CA.39

Let $S \subseteq \mathbb{R}^d$ *(a set of points). Let* $\overline{S} \subseteq \mathbb{R}^d$ *(a set of directions).*

$\text{conv}(S) + \text{cone}^0(\overline{S})$ *is the smallest convex set* $\begin{cases} \textit{containing all the points in } S \\ \textit{receding in all the directions in } \overline{S}. \end{cases}$

Theorem CA.40 (*Klee*)

Let $C \subseteq \mathbb{R}^d$.
If C *is a closed convex set containing no line* ***then*** $C = \text{conv}(\text{extr}(C)) + \text{cone}(\overrightarrow{extr}(C))$.

CA.3.7 Polyhedra

Definition CA.41

An (affine) ***hyperplane*** *is a set of the form* $H=\left\{x\in\mathbb{R}^d \,\middle|\, a\cdot x=\alpha\right\}$ *for given* $a\in\mathbb{R}^d$ *and* $\alpha\in\mathbb{R}$.
If $\alpha=0$, H *is a vectorial hyperplane.*

Such an hyperplane defines two ***half-spaces****:* $\boldsymbol{H}^{\geqslant}=\left\{x\in\mathbb{R}^d \,\middle|\, a\cdot x\geqslant\alpha\right\}$ *and* $\boldsymbol{H}^{\leqslant}=\left\{x\in\mathbb{R}^d \,\middle|\, a\cdot x\leqslant\alpha\right\}$.

Definition CA.42

A ***polyhedron*** *is an intersection of finitely many half-spaces.*
A ***polytope*** *is a bounded polyhedron.*

Remark CA.43

A polyhedron is a closed convex set. A polytope is a compact convex set.

Theorem CA.44 (*Minkowski-Weyl*)

Let $P\subseteq\mathbb{R}^d$ *such that* $P\neq\emptyset$.
P *is a polyhedron* ***if and only if*** P *is convex and finitely generated*
i.e. there exist two finite sets $E\subseteq\mathbb{R}^d$ *and* $D\subseteq\mathbb{R}^d$ *such that* $P=\mathrm{conv}(E)+\mathrm{cone}^0(D)$.

Remark CA.45

The minimal subsets E and D suitable to define polyhedron P in the previous property are known.
According to Klee's theorem, if P is a polyhedron, we have $P=\mathrm{conv}(\mathrm{extr}(P))+\mathrm{cone}^0(\overrightarrow{\mathrm{extr}}(P))$.
In particular, if P is a polytope, we have $P=\mathrm{conv}(\mathrm{extr}(P))$.

CA.3.8 Face and facets

Definition CA.46

Let $P\subseteq\mathbb{R}^d$ *be a polyhedron. Let* $F\subseteq\mathbb{R}^d$ *such that* $F\neq\emptyset$. *Let* $x\in\mathbb{R}^d$

F *is a* ***face*** *of* P *if and only if there exists an hyperplane* H *such that* $\begin{cases}H\cap P=F\\ P\subseteq H^{\geqslant}\text{ or }P\subseteq H^{\leqslant}\end{cases}$

F *is a* ***facet*** *of* P *if* F *is a face of* P *and* $\dim(F)=\dim(P)-1$.

x *is a* ***vertex*** *of* P *if* $\{x\}$ *is a 0-dimensional face of* P.

Remark CA.47

• In order to have a lattice structure on the set of the faces of a polyhedron, the empty set and the polyhedron itself are sometimes considered as faces.
• Faces can also be defined for convex sets rather than for polyhedron. Of course the convex definition coincides with the above definition in the particular case where the convex set is a polyhedron.
A face of a convex set C is a non-empty part of C, minimal for the inclusion, that can be removed without impairing the convexity of C. Note that removing a face of a polyhedron turns it into a convex set that is no more a polyhedron, since it is in particular no more a closed set.

Property CA.48

Let $P\subseteq\mathbb{R}^d$ *be a polyhedron. Let* $F\subseteq\mathbb{R}^d$ *such that* $F\neq P$.
If F *is a face of* P, ***then*** *there exists* F' *a facet of* P *such that* $F\subseteq F'$.

Property CA.49

Let $P\subseteq\mathbb{R}^d$ *be a polyhedron. Let* $x\in\mathbb{R}^d$. x *is a vertex of* P ***if and only if*** $x\in\mathrm{extr}(P)$.

CA.4 Useful properties for transposing facets

We recall here some definitions and properties that are used to prove Property 3.10.

Definition CA.50

Let $\mathcal{A}^1$ and $\mathcal{A}^2$ be two affine spaces. Let $\psi \in \mathcal{F}(\mathcal{A}^1, \mathcal{A}^2)$.
*ψ is an **affine map** between $\mathcal{A}^1$ and $\mathcal{A}^2$ if it preserves the affine combinations, i.e.*

$$\forall (\lambda_i)_{i\in I} \in \mathbb{R}^I,\ \forall (a_i)_{i\in I} \in (\mathcal{A}^1)^I,\ \sum_{i\in I} \lambda_i = 1 \Rightarrow \psi\Big(\sum_{i\in I} \lambda_i\, x_i\Big) = \sum_{i\in I} \lambda_i\, \psi(x_i)$$

Lemma CA.51

Let $\mathcal{A}^1$ and $\mathcal{A}^2$ be two affine spaces. Let $\psi \in \mathcal{F}(\mathcal{A}^1, \mathcal{A}^2)$ an affine map between these two spaces.
*(i) **If** $(s_i)_{i\in I} \in (\mathcal{A}^1)^I$ are not affinely independent **then** $\big(\psi(s_i)\big)_{i\in I}$ are also not affinely independent.*
*(ii) **If** C^1 is a convex set of $\mathcal{A}^1$, **then** $C^2 = \psi(C^1)$ is also a convex set **and** $\mathrm{extr}(C^2) \subseteq \psi\big(\mathrm{extr}(C^1)\big)$.*
*(iii) **If in addition** ψ is bijective, **then** $\mathrm{extr}(C^2) = \psi\big(\mathrm{extr}(C^1)\big)$.*

Proof: • Let $(s_i)_{i\in I}$ a family of points in $\mathcal{A}^1$. Let assume that $(s_i)_{i\in I}$ are not affinely independent.
By definition, there exists $i_0 \in I$ and $(\lambda_i)_{i\in I\setminus\{i_0\}} \in \mathbb{R}^{I\setminus\{i_0\}}$ such that :

$$\sum_{i\in I\setminus\{i_0\}} \lambda_i = 1 \quad \text{and} \quad \sum_{i\in I\setminus\{i_0\}} \lambda_i\, x_i = x_{i_0}.$$

Since ψ is an affine map, we then have:

$$\psi(x_{i_0}) = \psi\left(\sum_{i\in I\setminus\{i_0\}} \lambda_i\, x_i\right) = \sum_{i\in I\setminus\{i_0\}} \lambda_i\, \psi(x_i).$$

In other words $\psi(x_{i_0})$ is an affine combination of points $\big(\psi(s_i)\big)_{i\in I\setminus\{i_0\}}$.
Therefore, $\big(\psi(s_i)\big)_{i\in I}$ is not affinely independent. Hence, *(i)* is true.

• Let $(x^2, y^2) \in C^2 \times C^2$. Since $C^2 = \psi(C^1)$, there exists $(x^1, y^1) \in C^1 \times C^1$ such that $\begin{cases} \psi(x^1) = x^2 \\ \psi(y^1) = y^2 \end{cases}$.
Then, for any $\theta \in [0,1]$, we have:

$$\theta\, x^2 + (1-\theta)\, y^2 = \theta\, \psi(x^1) + (1-\theta)\, \psi(y^1) = \psi(\underbrace{\theta\, x^1 + (1-\theta)\, y^1}_{\in C^1 \text{ by } C^1 \text{ convexity}}) \in \psi(C^1) = C^2.$$

Thus C^2 is a convex set.

Let $z^2 \in \mathrm{extr}(C^2)$. Since $C^2 = \psi(C^1)$, there exists $z^1 \in C^1$ such that $\psi(z^1) = z^2$.
By contradiction, let us assume that $z^1 \notin \mathrm{extr}(C^1)$.
Then, by definition, there exist $(x^1, y^1) \in C^1 \times C^1$ and $\theta \in [0,1]$ such that $z^1 = \theta\, x^1 + (1-\theta)\, y^1$.
Then we have:

$$z^2 = \psi(z^1) = \psi(\theta\, x^1 + (1-\theta)\, y^1) = \psi(x^1) + (1-\theta)\psi(y^1)$$

Since $\psi(x^1) \in C^2$ and $\psi(y^1) \in C^2$, we deduce that z^2 is not an extreme point of C^2. A contradiction.
Then $z^1 \in \mathrm{extr}(C^1)$ and $z^2 \in \psi\big(\mathrm{extr}(C^1)\big)$. Therefore, we deduce that $\mathrm{extr}(C^2) \subseteq \psi\big(\mathrm{extr}(C^1)\big)$.
Finally, *(ii)* is true.

• In the case where ψ in bijective, ψ^{-1} is also an affine map.
Since $C^1 = \psi^{-1}(C^2)$, we deduce from *(ii)* that $\mathrm{extr}(C^1) \subseteq \psi^{-1}\big(\mathrm{extr}(C^2)\big)$.
Then we have $\psi\big(\mathrm{extr}\, C^1\big) \subseteq \psi \circ \psi^{-1}\big(\mathrm{extr}(C^2)\big) = \mathrm{extr}(C^2)$. Hence, *(iii)* is true. □

Remark CA.52

If ψ is not bijective, we have not necessarily $\psi(\text{extr}(C^1)) = \text{extr}(C^2)$.
For example, if C^1 is a square $ABCD$ and ψ the projection on (A, C), then $C^2 = [A, C]$. Extreme points of C^2, namely A and C, are images of extreme points of C^1 (A and C). Conversely, the other extreme points of C^1, namely B and D, are mapped to the middle of $[A, C]$ which is not extreme.

Corollary CA.53

Let $\mathcal{A}^1$ and $\mathcal{A}^2$ be two affine spaces. Let $\psi \in \mathcal{F}(\mathcal{A}^1, \mathcal{A}^2)$.
***If** ψ is an affine map between $\mathcal{A}^1$ and $\mathcal{A}^2$ **then** for any $S \subseteq \mathcal{A}^1$, $\dim(\psi(S)) \leqslant \dim(S)$.*
***If in addition** ψ is bijective, **then** for any $S \subseteq \mathcal{A}^1$, $\dim(\psi(S)) = \dim(S)$.*

Definition CA.54

Let $\mathcal{A}^1$ be an affine space and H be an hyperplane of $\mathcal{A}^1$.
Let H^+ and H^- be the two half-spaces defined by H.
*H is **valid** for $S \subseteq \mathcal{A}^1$ if $S \subseteq H^+$ or $S \subseteq H^-$.*

Lemma CA.55

Let $S \subseteq \mathbb{R}^d$, and $P = \text{conv}(S)$. Let $H \subseteq \mathbb{R}^d$ an hyperplane. H is valid for $P \Leftrightarrow$ H is valid for S.

Proof: By definition of an hyperplane, there exists an affine map $f \in \mathcal{F}(\mathcal{A}^1, \mathbb{R})$ s.t. $H = \{x \in \mathcal{A}^1 \mid f(x) = 0\}$.
Since $S \subseteq P$, H is valid for $P \Rightarrow H$ is valid for S. Let us show the reverse implication.

Let assume that H is valid for S. Without loss of generality, we assume that $S \subseteq H^{\leqslant} = \{x \in \mathcal{A}^1 \mid f(x) \leqslant 0\}$, even if f must be changed into $-f$. Let $x \in P$. Since $P = \text{conv}(S)$, there exist I a finite set, $(\lambda_i)_{i \in I} \in [0, 1]^I$, and $(s_i)_{i \in I} \in S^I$ such that $x = \sum_{i \in I} \lambda_i \, s_i$ and $\sum_{i \in I} \lambda_i = 1$. Then, since f is affine, we have:

$$f(x) = f\Big(\sum_{i \in I} \lambda_i \, s_i\Big) = \sum_{i \in I} \underbrace{\lambda_i}_{\geqslant 0} \underbrace{f(s_i)}_{\leqslant 0} \leqslant 0$$

Therefore, $x \in H^{\leqslant}$. Hence $P \subseteq H^{\leqslant}$, that is H is valid for P.

□

Property CA.56

Let $\mathcal{A}^1$ be an affine space and C be a convex set of $\mathcal{A}^1$. Let H be an hyperplane of $\mathcal{A}^1$ valid for C.
***If** F is the face of C defined by H, i.e. $F = C \cap H$, **then** $\text{extr}(F) = \text{extr}(C) \cap H$.*

Proof: Since F is a convex set and $F \subseteq C$, Lemma CA.32 ensures that $\text{extr}(C) \cap F \subseteq \text{extr}(F)$.
By definition of C, we have $\text{extr}(C) \cap F = \text{extr}(C) \cap (C \cap H) = \text{extr}(C) \cap H$.
Then we have $\text{extr}(C) \cap H \subseteq \text{extr}(F)$. Let us show the reverse inclusion.

By definition of an hyperplane, there exists an affine map $f \in \mathcal{F}(\mathcal{A}^1, \mathbb{R})$ s.t. $H = \{x \in \mathcal{A}^1 \mid f(x) = 0\}$.
Since H is valid for C, we assume that $C \subseteq H^{\leqslant} = \{x \in \mathcal{A}^1 \mid f(x) \leqslant 0\}$, even if f must be changed into $-f$.
Let $x \in \text{extr}(F)$. Necessarily $x \in H$. By contradiction, let us assume that $x \notin \text{extr}(C)$.
Then there exist $(u, v) \in C \times C$ and $\theta \in [0, 1]$ such that $x = \theta \, u + (1 - \theta) \, v$.
Since $u \in C \subseteq H^{\leqslant}$, we have $f(u) \leqslant 0$. Similarly, $f(v) \leqslant 0$.
Since f is affine, we also have $\theta \, f(u) + (1 - \theta) \, f(v) = f(\theta \, u + (1 - \theta) \, v) = f(x) = 0$.
We deduce that $f(u) = f(v) = 0$ that is $(u, v) \in H \times H$.
Then x is a convex combination of u and v, two points of $C \cap H = F$. A contradiction.
Therefore, $\text{extr}(F) \subseteq \text{extr}(C) \cap H$.

□

Corollary CA.57

Let $S \subseteq \mathbb{R}^d$, and $P = \text{conv}(S)$. Let $H \subseteq \mathbb{R}^d$ a valid hyperplane for S.
(i) $H \cap P \neq \emptyset \Leftrightarrow H \cap S \neq \emptyset$.
(ii) $\dim(H \cap P) = \dim(H \cap S)$.

Proof : • Since $S \subseteq P$, we have $H \cap S \subseteq H \cap P$, thus $H \cap S \neq \emptyset \Rightarrow H \cap P \neq \emptyset$.

Conversely, we have:
$H \cap P \neq \emptyset \Leftrightarrow \text{extr}(H \cap P) \neq \emptyset$ by Corollary CA.38*(i)*
$\Leftrightarrow H \cap \text{extr}(P) \neq \emptyset$ since $\text{extr}(H \cap P) = H \cap \text{extr}(P)$ by Property CA.56
$\Rightarrow H \cap S \neq \emptyset$ since $\text{extr}(P) \subseteq S$ according to Remark CA.31

• Since $S \subseteq P$, we have $H \cap S \subseteq H \cap P$ thus $\dim(H \cap S) \leqslant \dim(H \cap P)$.

Conversely, we have:
$\dim(H \cap P) = \dim(\text{extr}(H \cap P))$ by Corollary CA.38*(ii)*
$= \dim(H \cap \text{extr}(P))$ since $\text{extr}(H \cap P) = H \cap \text{extr}(P)$ by Property CA.56
$\leqslant \dim(H \cap S)$ since $\text{extr}(P) \subseteq S$ according to Remark CA.31

□

Notations

General notations

- *Sets of numbers*

$\mathbb{N}$	denotes the set of non-negative integers
$\mathbb{Z}$	denotes the set integers
$\mathbb{R}$	denotes the set of real numbers
$\mathbb{R}_+$	denotes the set of non-negative real numbers
$\mathbb{R}_+^*$	denotes the set of positive real numbers
$\overline{\mathbb{R}_+}$	denotes $\mathbb{R}_+ \cup \{+\infty\}$
$[a..b]$	denotes the set of integers i such that $a \leqslant i \leqslant b$ for any $(a,b) \in \mathbb{N}^2$, potentially $\emptyset$

- *Notations for a set X*

$\lvert X \rvert$	denotes the cardinality of X
$\overline{X}$	denotes the complement of the set X
$X \setminus Y$	denotes the relative complement of Y in X, for any set Y
$X \cap Y$	denotes the intersection of X and Y, for any set Y
$X \cup Y$	denotes the union of X and Y, for any set Y
$X \sqcup Y$	denotes the disjoint union of X and Y, for any set Y such that $X \cap Y \neq \emptyset$
$X \triangle Y$	denotes the symmetric difference X and Y, for any set Y, *i.e.* $(X \cup Y) \setminus (X \cap Y)$
$X/\sim$	denotes the quotient space of X by an equivalence relation $\sim$, that is the set of the equivalence classes for $\sim$
$X^<$	denotes $\{(i,j) \in X \times X \mid i < j\}$ if $(X, <)$ is an ordered set
$X^\leqslant$	denotes $\{(i,j) \in X \times X \mid i \leqslant j\}$ if $(X, <)$ is an ordered set
$X^{<<}$	denotes $\{(i,j,k) \in X \times X \times X \mid i < j < k\}$ if $(X, <)$ is an ordered set
$X : Y$	denotes $\{\{x,y\} \mid (x,y) \in X \times Y\}$, for any set Y such that $X \cap Y \neq \emptyset$
$\mathcal{P}(X)$	denotes the set of the X subsets
$\mathcal{P}^*(X)$	denotes the set of the non-empty X subsets
$\vec{\mathcal{P}}_2(X)$	denotes the set of oriented bi-partitions of X, *i.e.* $\{(A,B) \mid \{A,B\}$ is a partition of $X\}$
$\mathcal{F}(X,Y)$	denotes the set of the functions from X to Y

• *Notations for a function* $f \in \mathcal{F}(X,Y)$

$f_{/X'}$	denotes the restriction of f to the domain, for a subset $X' \subseteq X$
$f(A)$	denotes the image of a set $A \subseteq X$ under f, *i.e.* $\{f(x) \mid x \in A\}$
$f^{-1}(B)$	denotes the preimage of a set $B \subseteq Y$ under f, *i.e.* $\{x \in X \mid f(x) \in B\}$
$g \circ f$	denotes the composition of f and g for any $g \in \mathcal{F}(Y,Z)$, *i.e.* $\forall x \in X,\ g \circ f(x) = g(f(x))$

• *Notations for a real number* $t \in \mathbb{R}$

$[t]^+$	denotes the positive part of t *i.e.* $[t]^+ = \max(t, 0)$
$[t]^-$	denotes the negative part of t *i.e.* $[t]^- = \max(-t, 0)$
$\lfloor t \rfloor$	denotes the floor of t *i.e.* $\lfloor t \rfloor = \max\{k \in Z \mid k \leqslant t\}$
$\lceil t \rceil$	denotes the ceil of t *i.e.* $\lceil t \rceil = \min\{k \in Z \mid k \geqslant t\}$
$[t]$	denotes the integer part of t *i.e.* $[t] = \max\{k \in Z \mid \lvert k \rvert \leqslant \lvert t \rvert\}$

• *Notations for a vector* $x \in \mathbb{R}^n$

x_i	denotes the i^{th} component of x, for any $i \in [1..n]$
$x(S)$	denotes the sum of x components having an index in S, for any $S \subseteq [1..n]$, *i.e.* $x(S) = \sum_{i \in S} x_i$
$x \cdot y$	denotes the scalar product of x and $y \in \mathbb{R}^n$ *i.e.* $x \cdot y = \sum_{i=1}^{n} x_i\, y_i$
$x * y(S)$	denotes the sum $\sum_{i \in S} x_i\, y_i$ for any $y \in \mathbb{R}^n$ and $S \subseteq [1..n]$
$x_{/S}$	denotes the restriction of x to $S \subseteq [1..n]$, *i.e.* $x_{/S} = (x_j)_{j \in S}$

• *Linear algebra notations*

$\mathrm{vect}(S)$	denotes the vectorial space generated by a set of points $S \subseteq \mathbb{R}^n$
$\mathrm{aff}(S)$	denotes the affine space generated by a set of points $S \subseteq \mathbb{R}^n$
$\dim(S)$	denotes the dimension of the affine space generated by a set of points $S \subseteq \mathbb{R}^n$, *i.e.* $\dim(S) = \dim\Big(\mathrm{aff}(S)\Big)$ in general, and $\dim(S) = \dim\Big(\mathrm{vect}(S)\Big)$ if $0 \in S$

• *Convex analysis notations, for* $S \subseteq \mathbb{R}^n$

$\mathrm{conv}(S)$	denotes the convex hull of S
$\mathrm{cone}(S)$	denotes the convex cone generated by S
$\mathrm{cone}^0(S)$	denotes the convex cone generated by $S \cup \{0\}$
$\mathrm{extr}(C)$	denotes the set of extreme points of a convex set $C \subseteq \mathbb{R}^n$
$0^+(C)$	denotes the recession cone of a convex set $C \subseteq \mathbb{R}^n$
$\overrightarrow{\mathrm{extr}}(C)$	denotes the set of extreme direction of a convex set $C \subseteq \mathbb{R}^n$

• *Other general notations*

$\mathbb{I}$	denotes the vector $(1, 1, \ldots, 1) \in \mathbb{R}^n$
$\mathbb{I}_S$	denotes the indicator vector of a set $S \subseteq [1..n]$, *i.e.* $(\mathbb{I}_S)_i = 1 \Leftrightarrow i \in S$
$\mathbb{I}_{\{i\}}$	denotes then the ith vector of the canonical basis, for any $i \in [1..n]$

Scheduling notations

- *For a given instance*

J denotes the set of tasks

n denotes le number of tasks, *i.e.* $n=|J|$

d denotes the common due date

- *For a given instance, and for a given task $j\in J$ of this instance*

p_j denotes the processing time of a task $j\in J$

α_j denotes the unitary earliness penalty of a task $j\in J$

β_j denotes the unitary tardiness penalty of a task $j\in J$

$A(j)$ denotes the set of tasks having a larger α-ratio than j, *i.e.* $A(j)=\left\{i\in J \;\middle|\; \frac{\alpha_i}{p_i}>\frac{\alpha_j}{p_j}\right\}$

$\bar{A}(j)$ denotes the set of other tasks having a smaller α-ratio than j, *i.e.* $\bar{A}(j)=\left\{i\in J\setminus\{j\} \;\middle|\; \frac{\alpha_i}{p_i}\leqslant\frac{\alpha_j}{p_j}\right\}$

$B(j)$ denotes the set of tasks having a larger β-ratio than j, *i.e.* $B(j)=\left\{i\in J \;\middle|\; \frac{\beta_i}{p_i}>\frac{\beta_j}{p_j}\right\}$

$\bar{B}(j)$ denotes the set of other tasks having a smaller β-ratio than j, *i.e.* $\bar{B}(j)=\left\{i\in J\setminus\{j\} \;\middle|\; \frac{\beta_i}{p_i}\leqslant\frac{\beta_j}{p_j}\right\}$

- *For a given schedule $\mathcal{S}$*

$C_j(\mathcal{S})$ [1] denotes the completion time of a task $j\in J$

$E_j(\mathcal{S})$ [1] denotes the earliness of a task $j\in J$, *i.e.* $E_j=[d-C_j]^+$

$T_j(\mathcal{S})$ [1] denotes the tardiness of a task $j\in J$, *i.e.* $T_j=[C_j-d]^+$

$E(\mathcal{S})$ denotes the set of early tasks *i.e.* $E(\mathcal{S})=\{\,j\in[1..n]\,|\,C_j\leqslant d\,\}=\{\,j\in[1..n]\,|\,T_j=0\,\}$

$T(\mathcal{S})$ denotes the set of tardy tasks *i.e.* $T(\mathcal{S})=\{\,j\in[1..n]\,|\,C_j>d\,\}=\{\,j\in[1..n]\,|\,T_j>0\,\}$

Graph notations

- *For a given undirected graph $G=(V,E)$*

$\mathcal{C}y(G)$ denotes the set of the elementary cycles of G

$\mathrm{supp}(C)$ denotes the support of C for any cycle $C\in\mathcal{C}y(G)$

$N(u)$ denotes the neigborhood of a node $u\in V$, *i.e.* $N(u)=\{v\in V\setminus\{u\}\,|\,\{u,v\}\in E\}$

[1] The $\mathcal{S}$ will often be implicit.

Inequalities for F^3 and F^4

$$\forall (i,j) \in J^<,\ \ X_{i,j} \geqslant \delta_i - \delta_j \tag{X.1}$$
$$\forall (i,j) \in J^<,\ \ X_{i,j} \geqslant \delta_j - \delta_i \tag{X.2}$$
$$\forall (i,j) \in J^<,\ \ X_{i,j} \leqslant \delta_i + \delta_j \tag{X.3}$$
$$\forall (i,j) \in J^<,\ \ X_{i,j} \leqslant 2 - (\delta_i + \delta_j) \tag{X.4}$$

$$\forall j \in J,\ \ e_j \geqslant 0 \tag{1.5}$$
$$\forall j \in J,\ \ e_j \leqslant \delta_j\,(p(J) - p_j) \tag{1.6}$$
$$\forall j \in J,\ \ t_j \geqslant 0 \tag{1.7}$$
$$\forall j \in J,\ \ t_j \leqslant (1-\delta_j)\,p(J) \tag{1.8}$$

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

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

$$\forall j \in J,\ \ e'_j + p_j\,\delta_j \leqslant d - a \tag{1.15}$$
$$\sum_{j \in J} p_j\,\delta_j \leqslant d - a \tag{1.16}$$
$$a \geqslant 0 \tag{1.17}$$

$$\sum_{j \in J} \gamma_j = 1 \tag{1.18}$$
$$\forall j \in J,\ \ \delta_j \leqslant 1 - \gamma_j \tag{1.19}$$
$$\forall j \in J,\ \ t'_j \leqslant p_j + (1-\gamma_j)\,(p(J) - p_j) \tag{1.20}$$

$$\forall j \in J,\ \ a \leqslant p_j + (1-\gamma_j)\,d \tag{1.21}$$

$$\forall j \in J,\ \ b_j \geqslant 0 \tag{1.22}$$
$$\forall j \in J,\ \ b_j \leqslant a \tag{1.23}$$
$$\forall j \in J,\ \ b_j \leqslant \delta_j\, d \tag{1.24}$$
$$\forall j \in J,\ \ b_j \geqslant a - (1-\delta_j)\,d \tag{1.25}$$

Index of definitions

Index of polyhedra and formulations