République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A.Mira - BEJAIA

Faculté des Sciences Exactes
Département de Recherche Opérationnelle
Unité de recherche Modélisation et Optimisation des Systèmes (LaMOS)

THÈSE

EN VUE DE L'OBTENTION DU DIPLÔME DE DOCTORAT

Domaine : Mathématiques et Informatique **Filière :** Mathématiques Appliquées
Spécialité : Recherche Opérationnelle et Aide à la Décision

Présentée par

REZZAG Abdelkrim

Thème

Réduction des variables et application dans une méthode hybride pour la résolution des problèmes de programmation linéaire en nombres entiers

Soutenu le jeudi 08 Mai 2025 **Devant le jury composé de :**

Nom et Prénom	Grade		
Mr Mohammed Said RADJEF	**Professeur**	**Univ. de Béjaia**	**Président**
Mr Mohand Ouamer BIBI	**Professeur**	**Univ. de Béjaia**	**Rapporteur**
Mr Meziane AIDER	**Professeur**	**USTHB, Alger**	**Examinateur**
Mr Belkacem BRAHMI	**MCA**	**Univ. de Béjaia**	**Examinateur**
Mr Mohand BENTOBACHE	**Professeur**	**Univ. de Laghouat**	**Invité**

Année Universitaire : 2024/2025

⁕ Remerciements ⁕

Je remercie **DIEU**, le tout puissant, pour m'avoir accordé santé, courage et patience afin d'accomplir ce travail.

Je tiens à remercier mon directeur de thèse, Monsieur **BIBI Mohand Ouamer**, Professeur à l'Université de Béjaia, pour son encadrement précieux tout au long de cette recherche. Ses conseils éclairés, sa rigueur scientifique, et sa disponibilité constante ont été des éléments déterminants dans la réalisation de ce travail. Je remercie également sa patience et sa bienveillance, qui m'ont encouragé à surmonter les défis rencontrés et à progresser tant sur le plan académique que personnel. C'est un honneur d'avoir bénéficié de son expertise et de son accompagnement.

Je tiens à adresser mes sincères remerciements à Monsieur **RADJEF Mohammed Saïd**, Professeur à l'Université de Béjaia, pour l'honneur qu'il me fait en acceptant de présider le jury de ma thèse.

Je tiens à exprimer ma profonde gratitude à Monsieur **AIDER Meziane**, Professeur à l'Université des Sciences et de la Technologie Houari Boumediene d'Alger, à Monsieur **BENTOBACHE Mohand**, Professeur à l'Université de Laghouat, ainsi qu'à Monsieur **BRAHMI Belkacem**, Maître de conférences classe A à l'Université de Béjaia, pour avoir accepté de faire partie du jury de soutenance. Je les remercie sincèrement pour le temps précieux pour examiner ce travail.

Je tiens à remercier chaleureusement Monsieur **OUZIA Hacène**, Maître de Conférences à Sorbonne Université, pour m'avoir accueilli pendant un mois au sein du Laboratoire d'Informatique de Paris 6 (LIP6). Je suis reconnaissant pour les ressources mises à ma disposition ainsi que pour son soutien, qui m'ont permis de progresser significativement dans mes travaux de thèse.

J' exprime ma reconnaissance à mon collègue Monsieur **LAOUAR Abdelhek** pour ses précieux conseils et son orientation qui a été d'une grande valeur dans l'avancement de ce travail. J'exprime également toute ma gratitude à mes amis et collègues du LaMOS pour leur soutien, leur présence, leurs encouragements et leurs conseils, qui ont été une source précieuse de motivation et de réconfort tout au long de ce parcours.

Enfin, je tiens à adresser mes remerciements les plus sincères à ma mère, mon père, ma femme, mes frères et sœurs, dont le soutien indéfectible, la patience et l'amour inconditionnel ont été une source inestimable de force et d'inspiration. Leur présence à mes côtés a rendu possible l'accomplissement de ce travail, et je leur en serai éternellement reconnaissant.

Table des matières

Table des figures

Liste des Algorithmes

Liste des tableaux

Liste des symboles

$\mathbb{N}$	l'ensemble des entiers.
$\mathbb{Z}$	l'ensemble des entiers relatifs.
$\mathbb{R}$	l'ensemble des nombres réels.
x^T	la transposée de x.
$\mathbb{R}^n$	l'ensemble des n-vecteurs.
$\exists$	quantificateur existentiel (il existe).
$\forall$	quel que soit.
$\in$	le symbole d'appartenance.
$\notin$	le symbole de non appartenance.
$\lfloor \alpha \rfloor$	La partie entière inférieure de α.
$\lceil \alpha \rceil$	La partie entière supérieure de α.
$\Leftrightarrow$	le symbole d'équivalence.
$0_{\mathbb{R}^n}$	le vecteur nul à n composantes.
$\cup$	le symbole d'intersection.
$\cap$	le symbole de réunion.
$x := b$	remplacer x par b.

Liste des abréviations

PL	Programmation Linéaire
PLvb	Programmation Linéaire à variables bornées
PLNE	Programmation Linéaire en Nombres Entiers
PLNB	Programmation Linéaire en Nombres Binaires
Pr	Problème réduit
PR	Problème relaxé
SRS	Solution Réalisable de Support
SRB	Solution Réalisable de Base

Liste des contributions

Dans le cadre de cette thèse, nous avons réalisé les contributions scientifiques suivantes :

— **Article**

1. A. Rezzag, M.O. Bibi and A. Laouar. Reduction of bounded variables in integer linear programming problems. International Journal of Mathematics in Operational Research, 30(2):251-265, 2025.

— **Conférences et Séminaires**

1. A. Rezzag. Procédure de presolving en programmation linéaire. Séminaire Mathématique de Béjaia, Unité de recherche LaMOS (Modélisation et Optimisation des systémes). Université de Béjaia, Algérie, octobre 2022.
2. A. Rezzag et M.O. Bibi. Procédure de presolving en programmation linéaire en nombres binaires. Colloque International Méthode et Outils d'Aide à la Décision MOAD'2022. Université de Béjaia, Algérie, novembre 2022.
3. A. Rezzag and M.O. Bibi. Variable Reduction for ILP Problems : Method and Analysis wtih Gurobi. The first National Conference on Applied Mathematics, Statistics and Applications NCAMSA'24. University of Batna 2, Algeria, november 2024.
4. A. Rezzag, M.O. Bibi and H. Ouzia. Rounding Heuristic for Integer Linear Programs. The 17th African Conference on Research in Computer Science and Applied Mathematics - Digital Sciences in Africa CARI'2024. University of Bejaia, Algeria, november 2024.
5. A. Rezzag, M.O. Bibi. Reduction of the bounds for ILP problems. The 4th National Conference of Mathematics and Applications CNMA - 2024. University center of Mila, Algeria, december 2024.

Introduction générale

L'optimisation joue un rôle fondamental dans divers domaines scientifiques et pratiques, notamment la logistique, la planification, la finance, et les réseaux, etc. Elle consiste à trouver les meilleures solutions possibles à des problèmes complexes soumis à des contraintes, qu'elles soient économiques, techniques ou temporelles. Parmi les techniques d'optimisation, la Programmation Linéaire (PL) et la Programmation Linéaire en Nombres Entiers (PLNE) s'imposent comme un outil essentiel pour modéliser et résoudre de nombreux problèmes réels. Ces techniques combinent rigueur mathématique et adaptabilité pratique, offrant des solutions optimales ou suboptimales dans des contextes variés.

La PL est introduite par Dantzig avec la méthode du simplexe [25], elle est devenue un pilier pour résoudre des problèmes où l'objectif et les contraintes sont linéaires. La théorie a été enrichie par des avancées comme le méthode adaptée [7, 8, 9, 10, 17, 18, 29, 33, 40]. Ces approches sont abondamment utilisées dans des domaines variés, comme la finance [20].

La PLNE constitue une extension cruciale de la PL, introduisant des contraintes d'intégralité sur certaines ou toutes les variables. Cependant, la résolution de problèmes de PLNE présente plusieurs défis, car elle est classée comme un problème NP-difficile. Les méthodes classiques incluent le Branch-and-Bound [6], les coupes de Gomory [37], et plus récemment des approches hybrides combinant ces algorithmes avec des heuristiques [18, 22].

Avec l'augmentation de la taille et de la complexité des problèmes de PLNE, il s'avère important d'adopter des techniques permettant de réduire les dimensions des problèmes tout en maintenant leurs propriétés fondamentales. C'est dans ce contexte que le presolving intervient. Il s'agit d'une phase de prétraitement visant à simplifier les problèmes mathématiques avant l'application des algorithmes de résolution. En éliminant les variables et contraintes redondantes et en resserrant les bornes des variables [3, 4, 5, 54, 64, 67, 68, 71, 72, 73], le presolving améliore considérablement l'efficacité des solveurs modernes comme CPLEX et Gurobi.

Enfin, l'heuristique d'arrondi est un outil complémentaire aux méthodes exactes dans la PLNE. Elle consiste à résoudre la relaxation continue du problème, puis à ajuster les solutions fractionnaires pour obtenir des solutions entières approximatives [2, 12, 18, 22, 43, 65]. Bien que ces heuristiques ne garantissent pas l'optimalité, elles sont souvent utilisées dans des contextes où le temps de calcul est critique.

Cette thèse s'inscrit dans ce cadre et propose une étude approfondie des techniques de presolving pour les problèmes de PLNE à variables bornées. L'objectif principal est de développer et d'analyser

des méthodologies qui permettent d'améliorer les performances des solveurs, tout en garantissant la qualité des solutions obtenues. Une approche heuristique est également explorée pour trouver des solutions suboptimales de bonne qualité dans des temps de calcul réduits.

Ainsi, cette thèse propose des améliorations aux techniques de presolving appliquées aux problèmes de PLNE, en particulier lorsque les variables de décision sont bornées. Plus précisément, elle est composée des parties suivantes :

1. Étude des bases mathématiques et algorithmiques : Cette partie présente des fondements de l'algèbre linéaire et des méthodes de résolution en programmation linéaire et en programmation linéaire en nombres entiers, notamment les méthodes du simplexe, branch-and-bound, et les coupes de Gomory.
2. Analyse approfondie du presolving : Cette deuxième partie est consacrée à une classification des différentes techniques de presolving, incluant des approches sur les contraintes, les bornes des variables, et les structures spécifiques des problèmes (variables binaires, non négatives, ou bornées).
3. Proposition d'une procédure innovante pour le presolving dans les problèmes de PLNE à variables bornées : Une approche méthodologique détaillée, appuyée par des résultats expérimentaux, permettant de réduire la complexité des problèmes tout en conservant leurs propriétés.
4. Développement d'une méthode heuristique d'arrondissement : Une méthode est élaborée pour obtenir des solutions suboptimales de bonne qualité, intégrant les procédures de presolving pour accélérer les calculs.

La thèse est composée d'une introduction générale, de quatre chapitres et d'une conclusion générale, détaillant progressivement les fondements, les méthodologies et les contributions de cette thèse :

— **Chapitre 1 : Préliminaires sur l'algèbre linéaire**

Ce chapitre présente les notions fondamentales de l'algèbre linéaire, indispensables pour comprendre les techniques d'optimisation. Les concepts clés tels que les espaces vectoriels, les matrices, et les systèmes linéaires sont exposés.

— **Chapitre 2 : Programmation Linéaire et Programmation Linéaire en Nombres Entiers**

Dans ce chapitre, les bases théoriques et les méthodes classiques de résolution de la programmation linéaire, telles que le simplexe, le support et la méthode adaptée avec un pas simple sont présentées. Pour la programmation linéaire en nombres entiers, les méthodes branch-and-bound, les coupes de Gomory, et l'heuristique d'arrondissement ont été passées en revue.

— **Chapitre 3 : Procédure de Presolving**

Ce chapitre constitue le cœur de la thèse. Il débute par une présentation du problème réduit, puis explore des méthodes simples de presolving. Les techniques appliquées aux contraintes, aux bornes des variables, et aux problèmes spécifiques (variables non négatives, binaires ou bornées) sont analysées en détail. Les propositions méthodologiques issues de notre travail sont également introduites ici.

— **Chapitre 4 : Méthode Heuristique d'Arrondissement et Presolving**

Ce dernier chapitre propose une méthode heuristique d'arrondissement intégrant les techniques de presolving pour obtenir des solutions suboptimales de bonne qualité. Une attention particulière est accordée à l'analyse des performances et des résultats obtenus.

Les travaux présentés dans cette thèse contribuent au domaine de l'optimisation, en mettant en lumière l'importance du presolving pour améliorer les performances des solveurs et réduire les difficultés des problèmes complexes. En proposant des méthodologies adaptées à des classes spécifiques de problèmes de PLNE et en développant des approches heuristiques, cette thèse ouvre la voie à de nouvelles perspectives pour la recherche et les applications pratiques. Les résultats obtenus pourraient notamment être étendus à d'autres types de problèmes d'optimisation, renforçant ainsi l'efficacité des outils actuels.

Chapitre 1

Préliminaires sur l'algèbre linéaire

Sommaire

1.1 Introduction

Au cours de ce chapitre, nous rappelons les concepts fondamentaux relatifs aux espaces vectoriels, aux matrices et aux vecteurs. Nous nous pencherons également sur les systèmes d'équations linéaires et les notions fondamentales de la convexité. Ces notions s'avèrent essentielles pour appréhender les méthodes et les résultats exposés tout au long de cette thèse.

1.2 Espaces vectoriels

Soit E un ensemble non vide. L'ensemble E est appelé un espace vectoriel sur un corps K (généralement $\mathbb{R}$) si les conditions suivantes sont satisfaites pour tous les éléments $x, y \in E$ et tous α, β de K :

1. E est équipé d'une opération interne notée $+$, de sorte que $(E,+)$ forme un groupe commutatif ;
2. E est doté d'une opération externe $K \times E \to E$ notée $(\alpha, x) \mapsto \alpha x$, vérifiant les propriétés suivantes :

$$\alpha(x+y) = \alpha x + \alpha y,$$
$$(\alpha+\beta)x = \alpha x + \beta x,$$
$$(\alpha\beta)x = \alpha(\beta x),$$
$$1 \cdot x = x.$$

Un ensemble qui satisfait ces conditions est ainsi appelé un espace vectoriel. Les éléments de E sont appelés des vecteurs. L'espace vectoriel le plus utilisé dans la suite de la thèse est l'ensemble $\mathbb{R}^n$.

1.2.1 Combinaison linéaire

Considérons $\{a_1, a_2, \ldots, a_k\}$ un ensemble de k vecteurs dans $\mathbb{R}^m$. Une combinaison linéaire de ces vecteurs prend la forme suivante :

$$\sum_{j=1}^{k} a_j x_j, \tag{1.1}$$

où x_j, $j = \overline{1,k}$, sont des scalaires réels. Ces scalaires sont les coefficients de la combinaison linéaire.

L'expression (1.1) peut également être formulée sous forme du produit d'une matrice A et d'un vecteur x :

$$\sum_{j=1}^{k} a_j x_j = (a_1 \ , \ a_2 \ , \ \ldots \ , \ a_k) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} = Ax, \tag{1.2}$$

où x_j est la $j^{\text{ème}}$ composante du vecteur-colonne x et $A = (a_j, j = \overline{1,k})$, a_j étant la $j^{\text{ème}}$ colonne de la matrice A.

Une combinaison linéaire est dite triviale si tous les coefficients sont égaux à zéro ($x_j = 0,\ j = \overline{1,k}$). En revanche, si au moins l'un des coefficients diffère de zéro, la combinaison linéaire nulle est dite non triviale.

1.2.2 Indépendance linéaire des vecteurs

Un ensemble fini de vecteurs $\{a_1, a_2, \ldots, a_k\}$ dans $\mathbb{R}^m$ est dit linéairement indépendant si :

$$\sum_{j=1}^{k} a_j x_j = 0 \Rightarrow x_j = 0, \quad \forall j = \overline{1,k}.$$

En d'autres termes, aucun vecteur de cet ensemble ne peut être représenté comme une combinaison linéaire des autres.

Si l'ensemble n'est pas linéairement indépendant, on dit qu'il est linéairement dépendant, signifiant qu'il existe des coefficients $x_1, x_2, \ldots, x_k$, non tous nuls, tels que :

$$\sum_{j=1}^{k} a_j x_j = 0, \quad \exists j_0 \in \{1, \ldots, k\}, \quad x_{j_0} \neq 0.$$

Cela implique qu'au moins un vecteur s'écrit comme une combinaison linéaire des autres.

1.2.3 Bases d'un espace vectoriel

Une base d'un espace vectoriel E sur $\mathbb{R}$ de dimension n, est un ensemble de n vecteurs de E, noté $B = \{v_1, v_2, \ldots, v_n\}$, vérifiant les deux conditions suivantes :

1. Tout vecteur x de E peut s'exprimer comme combinaison linéaire des vecteurs de B,
2. Les vecteurs de B sont linéairement indépendants.

1.2.4 Espace vectoriel normé

Considérons une fonction $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}^+$, appelée norme, qui assigne à chaque vecteur $x \in \mathbb{R}^n$ un scalaire réel non négatif, noté $\|x\|$. Pour tout $x, y \in \mathbb{R}^n$ et tout scalaire $\alpha \in \mathbb{R}$, cette fonction satisfait les propriétés suivantes :

- **Vecteur nul** : $\|x\| = 0 \Leftrightarrow x = 0$;
- **Homogénéité absolue** : $\|\alpha \cdot x\| = |\alpha| \cdot \|x\|$;
- **Inégalité triangulaire** : $\|x + y\| \leq \|x\| + \|y\|$.

Un espace vectoriel muni d'une norme est appelé un espace vectoriel normé.

1.2.5 Boule ouverte et Voisinage

1.2.5.1 Boule ouverte

Dans un espace normé $\mathbb{R}^n$, une boule ouverte de centre x^0 et de rayon $r > 0$ est définie comme l'ensemble $B_r(x^0)$, composé de tous les vecteurs x tels que la norme de la différence $\|x - x^0\|$ est strictement inférieure à r :

$$B_r(x^0) = \{x \in \mathbb{R}^n \ : \ \|x - x^0\| < r\}. \tag{1.3}$$

1.2.5.2 Voisinage

Soit $\mathbb{R}^n$ un espace vectoriel normé, et $x \in \mathbb{R}^n$. Un voisinage de x est tout sous-ensemble $V(x)$ de $\mathbb{R}^n$ tel qu'il existe un nombre réel positif $r > 0$ pour lequel $B_r(x) \subseteq V(x)$.

1.3 Matrices et vecteurs

Une matrice, que nous noterons A, est un tableau rectangulaire de nombres réels, généralement représenté sous la forme :

$$A = A(I,J) = (a_{ij},\ i \in I,\ j \in J) = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \tag{1.4}$$

où $I = \{1,\ldots,m\}$ est un ensemble fini non vide de numéros de lignes, et $J = \{1,\ldots,n\}$ est un ensemble fini non vide de numéros de colonnes. La matrice A a donc m lignes et n colonnes, ce qui donne une matrice d'ordre $m \times n$. Chaque élément de la matrice est noté a_{ij}, $i \in I$, $j \in J$.

Pour des calculs pratiques, nous pouvons également représenter la matrice A sous forme de colonnes ou de lignes :

$$A = (a_1\ ,\ \ldots\ ,\ a_j\ ,\ \ldots\ ,\ a_n) = \begin{pmatrix} A_1^T \\ \vdots \\ A_i^T \\ \vdots \\ A_m^T \end{pmatrix}, \tag{1.5}$$

où

$$a_j = A(I,j) = \begin{pmatrix} a_{1j} \\ \vdots \\ a_{ij} \\ \vdots \\ a_{mj} \end{pmatrix}, \tag{1.6}$$

est le $j^{\text{ème}}$ vecteur-colonne de la matrice A, de dimension m, considéré comme une matrice d'ordre $m \times 1$. De même,

$$A_i^T = A(i,J) = \left(a_{i1}\ ,\ \ldots\ ,\ a_{ij}\ ,\ \ldots\ ,\ a_{in}\right), \tag{1.7}$$

est le $i^{\text{ème}}$ vecteur-ligne de la matrice A, de dimension n, considéré comme une matrice d'ordre $1 \times n$.

Remarque 1.1. Nous introduisons également le symbole $()^T$ pour la transposition. En d'autres termes, si A est une matrice d'ordre $m \times n$, A^T est sa matrice transposée d'ordre $n \times m$, avec

$$A^T = A^T(J,I) = \left(a_{ji},\ j \in J,\ i \in I\right). \tag{1.8}$$

Remarque 1.2. Chaque vecteur x, noté $x = x(J) = \left(x_j,\ j \in J\right)$, sera ainsi considéré comme un vecteur-colonne, tandis que le vecteur-ligne sera noté x^T.

Définition 1.1. L'ensemble des matrices d'ordre $m \times n$ avec des coefficients dans $\mathbb{R}$ est noté $\mathscr{M}_{m,n}(\mathbb{R})$.

Définition 1.2 (Matrice carrée)**.** Si le nombre de lignes m est égal au nombre de colonnes n, alors A est dite matrice carrée d'ordre n, notée $A \in \mathscr{M}_n(\mathbb{R})$.

Définition 1.3 (Matrice diagonale)**.** Une matrice diagonale est une matrice carrée où tous les éléments en dehors de la diagonale sont nuls ($a_{ij} = 0$, lorsque $i \neq j$), et elle est notée

$$A = \operatorname{diag}\left(a_{11}\ ,\ \ldots\ ,\ a_{jj}\ ,\ \ldots\ ,\ a_{nn}\right) = \begin{pmatrix} a_{11} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & a_{jj} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & a_{nn} \end{pmatrix}.$$

Définition 1.4 (Matrice identité)**.** La matrice identité I_n d'ordre n est une matrice diagonale où tous les éléments de la diagonale principale sont égaux à 1.

$$I_n = \operatorname{diag}(1\ ,\ 1\ ,\ \ldots\ ,\ 1) = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}.$$

Définition 1.5 (Matrice triangulaire)**.** On dit que la matrice A est triangulaire supérieure (respectivement triangulaire inférieure) si $a_{ij} = 0$ pour $i > j$ (respectivement $i < j$).

On dit que la matrice A est strictement triangulaire supérieure (respectivement strictement triangulaire inférieure) si $a_{ij} = 0$ pour $i \geq j$ (respectivement $i \leq j$).

1.3.1 Opérations sur les matrices

Les opérations sur les matrices sont des transformations algébriques fondamentales qui permettent de combiner et de transformer des matrices. Ces opérations incluent l'addition, la multiplication par un scalaire et la multiplication matricielle.

Soient $A = (a_{ij},\ 1 \leq i \leq m,\ 1 \leq j \leq n)$ et $B = (b_{ij},\ 1 \leq i \leq m,\ 1 \leq j \leq n)$ deux matrices de $\mathscr{M}_{m,n}(\mathbb{R})$, soit $C = (c_{jk},\ 1 \leq j \leq n,\ 1 \leq k \leq p) \in \mathscr{M}_{n,p}(\mathbb{R})$ et $\lambda \in \mathbb{R}$. On définit alors les opérations suivantes :

— **Addition de matrices :** L'addition de A et B, notée $A+B$, est effectuée en additionnant les éléments correspondants :

$$A+B = (a_{ij}+b_{ij},\ 1 \leq i \leq m,\ 1 \leq j \leq n) \in \mathscr{M}_{m,n}(\mathbb{R}).$$

— **Multiplication par un scalaire :** La multiplication de la matrice A par le scalaire λ, notée λA, consiste à multiplier chaque élément de la matrice par λ :

$$\lambda A = (\lambda a_{ij},\ 1 \leq i \leq m,\ 1 \leq j \leq n) \in \mathscr{M}_{m,n}(\mathbb{R}).$$

— **Produit de deux matrices :** On appelle produit de la matrice A par la matrice C, notée $P = AC$, la matrice ayant le même nombre de lignes que A et le même nombre de colonnes que C, où chaque élément p_{ik} de cette matrice est obtenu en effectuant le produit scalaire de la $i^{\text{ème}}$ ligne de A par la $k^{\text{ème}}$ colonne de C :

$$P = AC = (p_{ik} = \sum_{j=1}^{n} a_{ij}c_{jk},\ 1 \leq i \leq m,\ 1 \leq k \leq p) \in \mathscr{M}_{m,p}(\mathbb{R}).$$

1.3.2 Matrices et vecteurs partitionnés

Les matrices et les vecteurs partitionnés permettent une manipulation efficace des produits matriciels, et ce, en utilisant le produit par blocs. Par exemple, si nous avons une matrice A et un vecteur-colonne x que nous avons partitionnés de manière judicieuse, le produit Ax est alors qualifié de multiplication par blocs. Plus précisément, si

$$A = (A_1\ ,\ A_2), \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

nous pouvons exprimer le produit Ax comme suit :

$$Ax = (A_1\ ,\ A_2)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = A_1x_1 + A_2x_2.$$

De manière similaire, pour

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix},$$

l'équation $Ax = b$ peut être réécrite de la manière suivante :

$$\begin{cases} A_{11}x_1 + A_{12}x_2 = b_1, \\ A_{21}x_1 + A_{22}x_2 = b_2. \end{cases}$$

La partition d'une matrice peut se faire de manière arbitraire. Par exemple, si $A = A(I,J) = (a_j,\ j \in J)$ est une matrice d'ordre $m \times n$, et si J_B et J_N sont deux sous-ensembles arbitraires de J tels que

$$m < n, \quad |J_B| = |I| = m, \quad J_B \cup J_N = J, \quad J_B \cap J_N = \emptyset,$$

alors nous pouvons partitionner A comme suit :

$$A = (A_B\ ,\ A_N),$$

avec $A_B = A(I,J_B) = (a_j,\ j \in J_B)$ et $A_N = A(I,J_N) = (a_j,\ j \in J_N)$.

Si

$$x = x(J) = \begin{pmatrix} x_B \\ x_N \end{pmatrix},$$

où $x_B = x(J_B) = (x_j,\ j \in J_B)$ et $x_N = x(J_N) = (x_j,\ j \in J_N)$, alors le produit Ax peut s'exprimer comme suit :

$$\begin{aligned} Ax = \sum_{j \in J} a_j x_j &= \sum_{j \in J_B} a_j x_j + \sum_{j \in J_N} a_j x_j \\ &= A(I,J_B)x(J_B) + A(I,J_N)x(J_N) \\ &= A_B x_B + A_N x_N. \end{aligned}$$

1.3.3 Déterminant d'une matrice

Le déterminant d'une matrice carrée A d'ordre n constitue une mesure numérique fondamentale associée à cette matrice, souvent utilisée pour évaluer l'inversibilité de celle-ci et pour résoudre des systèmes d'équations linéaires et des problèmes de programmation linéaire. Le déterminant de A, noté $\det(A)$ ou $|A|$, est défini de manière récurrente selon les cas suivants :

1. **Cas de base :** Pour une matrice 1×1, le déterminant est simplement égal à l'unique élément de la matrice :

   $$\det([a]) = a.$$

2. **Cas général (récurrent) :** Pour une matrice A d'ordre $n > 1$, le déterminant est calculé à l'aide de la formule suivante, pour j fixé, $1 \leq j \leq n$, on a :

   $$\det(A) = \sum_{i=1}^{n} (-1)^{i+j} \cdot a_{ij} \cdot \det(A_{ij}),$$

   où a_{ij} est l'élément de la matrice situé à la ligne i et à la colonne j, et A_{ij} est la sous-matrice obtenue en supprimant la ligne i et la colonne j de A. C'est le développement de A par rapport à une certaine colonne j de A.

1.3.4 Inverse d'une matrice

Définition 1.6. Une matrice carrée A d'ordre n est dite inversible (ou régulière, ou non singulière), s'il existe une matrice carrée B d'ordre n telle que $AB = BA = I_n$. Lorsque cette matrice B existe, elle est unique et elle est appelée l'inverse de A, noté $B = A^{-1}$.

Remarque 1.3. Si $\det(A) \neq 0$, la matrice admet une matrice inverse, et si $\det(A) = 0$, elle est dite singulière et n'a pas de matrice inverse.

1.3.5 Noyau et image d'un matrice

Considérons une matrice $A \in \mathscr{M}_{m,n}(\mathbb{R})$.

1.3.5.1 Noyau d'une matrice

Le noyau de A, noté $N(A)$, est l'ensemble des vecteurs $x \in \mathbb{R}^n$ tels que

$$N(A) = \{x \in \mathbb{R}^n \ : \ Ax = 0\}. \tag{1.9}$$

Le noyau d'une matrice est un sous-espace vectoriel de $\mathbb{R}^n$.

1.3.5.2 Image d'une matrice

L'image de A, noté $R(A)$, est l'ensemble des vecteurs y dans $\mathbb{R}^m$ tels que :

$$R(A) = \{y \in \mathbb{R}^m \ : \ y = Ax \ , \ \forall x \in \mathbb{R}^n\}. \tag{1.10}$$

L'image d'une matrice est un sous-espace vectoriel de $\mathbb{R}^m$.

1.3.6 Rang d'une matrice

Le rang d'une matrice A d'ordre $m \times n$, noté $rang(A)$, représente le nombre maximal de colonnes ou de lignes linéairement indépendantes dans la matrice. On a alors

$$rang(A) \leq \min\{m,n\}. \tag{1.11}$$

1.4 Systèmes d'équations linéaires

1.4.1 Forme générale

Soient m et n deux nombres entiers. Un système de m équations linéaires à n inconnus $x_1, \ x_2, \ \ldots, \ x_n$ est formulé comme suit :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \qquad\qquad\qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m, \end{cases} \tag{1.12}$$

où les coefficients a_{ij} sont des réels, et les nombres $b_1,\ b_2,\ \ldots,\ b_m$ sont appelés les membres libres du système (1.12) ou les seconds membres. Ce système peut être représenté sous la forme matricielle comme suit :

$$Ax = b, \tag{1.13}$$

où

$$A = (a_{ij},\ 1 \leq i \leq m,\ 1 \leq j \leq n), \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

La matrice augmentée $\bar{A} = (A|b)$ est créée en ajoutant le vecteur b à la matrice A. On a ainsi

$$rang(A) \leq rang(\bar{A}).$$

Définition 1.7. Le système linéaire (1.13) est dit de rang complet en lignes si $rang(A) = m$, avec $m \leq n$, et de rang complet en colonnes si $rang(A) = n$, avec $m \geq n$.

Définition 1.8 (Système homogène)**.** Un système est dit homogène si le vecteur des membres libres b est nul.

1.4.2 Opérations élémentaires sur les lignes d'un système linéaire

Considérons un système linéaire de m équations et n inconnues. Les opérations élémentaires sur les lignes i et j de ce système sont définies comme suit (en supposant $i \neq j$) :

- $L_i \leftrightarrow L_j$: échange de deux lignes.
- $L_i \leftarrow \alpha L_i$ (avec $\alpha \neq 0$) : Remplacement d'une ligne par son produit par un nombre réel non nul α.
- $L_i \leftarrow L_i + \beta L_j$: Remplacement d'une ligne par sa somme avec une autre ligne, multipliée par un nombre réel β.
- $L_i \leftarrow \alpha L_i + \beta L_j$ (avec $\alpha \neq 0$) : Combinaison linéaire des deux opérations précédentes.

1.4.3 Existence et le nombre de solutions

1.4.3.1 Solution du système

Définition 1.9. Tout vecteur-colonne x contenant les valeurs des inconnues et satisfaisant les équations du système (1.12) est appelé solution du système linéaire.

Théorème 1.1. *Le système* (1.13) *a une solution si et seulement si le vecteur* $b \in R(A)$*, c'est-à-dire* $rang(A) = rang(\bar{A})$*. Tout système linéaire homogène possède au moins la solution triviale* $x = 0$*.*

Définition 1.10 (Système compatible)**.** Le système est dit compatible s'il possède au moins une solution. Sinon, il est dit incompatible ou impossible, lorsque $rang(A) < rang(\bar{A})$.

1.4.3.2 équivalence des systèmes linéaires

Deux systèmes linéaires sont dits équivalents s'ils admettent les mêmes solutions. L'équivalence des systèmes linéaires est un concept important pour la résolution de systèmes d'équations linéaires. Ainsi, il est important de noter que les opérations élémentaires ci-dessus ne modifient pas l'ensemble des solutions du système. En d'autres termes, un système obtenu en appliquant ces opérations est équivalent au système originel, c'est-à-dire que les deux systèmes ont les mêmes solutions.

1.4.3.3 Nombre de solutions

La condition d'existence et le nombre de solutions dépendent du nombre d'équations et du nombre d'inconnues dans le système (1.13).

Lemme 1.1. *Soit le système* (1.13) *avec* $rang(A) = m$*, il admet :*

- ***Une seule solution*** *: cela se produit lorsque* $m = n$*.*
- ***Une infinité de solutions*** *: cela produit lorsque* $m < n$*.*

Lemme 1.2. *Si* $rang(A) < m$ *et* $n < m$*, alors ce système n'admet aucune solution.*

1.4.3.4 Solution basique

Soit $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$ un vecteur partitionné qui est la solution du système (1.13), où $rang(A) = m < n$, avec $x_B \in \mathbb{R}^m$. Alors x est dit solution basique si $x_N = 0$ et si les vecteurs composant la sous-matrice carrée A_B sont linéairement indépendants.

Une solution basique est dite non dégénérée si

$$x_j \neq 0,\ \forall j \in J_B,\ \text{ avec } x_B = A_B^{-1} b.$$

Les solutions basiques réalisables sont souvent utilisées dans le cadre de la programmation linéaire pour trouver les solutions optimales.

1.4.4 Décomposition *LU*

La décomposition LU d'une matrice $A \in \mathscr{M}_n(\mathbb{R})$, avec $\det(A) \neq 0$, consiste à la factoriser en deux matrices, une matrice triangulaire inférieure L avec des uns sur la diagonale et une matrice triangulaire supérieure U, telles que $A = LU$. Cette décomposition est utilisée pour résoudre le système (1.13), lorsque $m = n$.

En partant de $A = LU$, le système (1.13) peut être réécrit sous la forme $L(Ux) = b$. En posant $Ux = y$, on procède à la résolution en deux étapes. D'abord, on résout le système triangulaire inférieur $Ly = b$, puis le système triangulaire supérieur $Ux = y$.

1.5 Notions sur la convexité

La convexité est un concept fondamental en mathématiques, en particulier dans le domaine de l'optimisation. Elle se révèle être un outil indispensable, offrant des conditions d'optimalité à la fois nécessaires et suffisantes.

1.5.1 Ensembles convexes

Un ensemble E de $\mathbb{R}^n$ est dit convexe si

$$\forall x, y \in E, \quad \forall \lambda \in [0;1] \Rightarrow \lambda x + (1-\lambda) y \in E. \tag{1.14}$$

En d'autres termes, le segment de droite reliant deux points quelconques de l'ensemble E est entièrement contenu dans E.

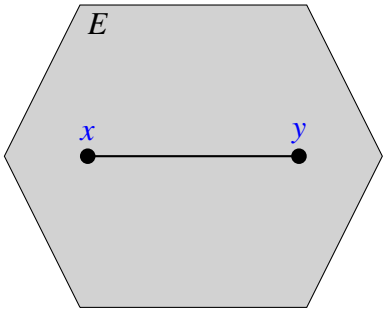

FIGURE 1.1 – Ensemble convexe

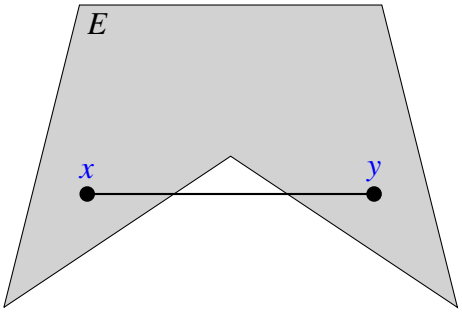

FIGURE 1.2 – Ensemble non convexe

Définition 1.11 (Combinaison Convexe)**.** Considérons $x^1, x^2, \ldots, x^k$ des vecteurs de $\mathbb{R}^n$. Un vecteur $x \in \mathbb{R}^n$ est une combinaison convexe de ces vecteurs, s'il existe des coefficients réels $\alpha_1, \alpha_2, ..., \alpha_k$ tels que :

$$x = \sum_{j=1}^{k} \alpha_j x^j, \text{ avec } \sum_{j=1}^{k} \alpha_j = 1, \quad \alpha_j \geq 0, \quad j = \overline{1,k}. \tag{1.15}$$

1.5.2 Propriétés des ensembles convexes

Propriété 1.1. *Si E_1, E_2 sont deux ensembles convexes dans $\mathbb{R}^n$ et α_1, α_2 sont deux réels, alors l'ensemble*

$$E_3 = \alpha_1 E_1 + \alpha_2 E_2 = \{x^3 \in \mathbb{R}^n \ : \ x^3 = \alpha_1 x^1 + \alpha_2 x^2, \ x^1 \in E_1, \ x^2 \in E_2\},$$

est également un ensemble convexe dans $\mathbb{R}^n$.

Propriété 1.2. *Si $\{E_j\}_{j=\overline{1,k}}$ est une famille d'ensembles convexes dans $\mathbb{R}^n$, alors l'intersection $\bigcap_{j=1}^{k} E_j$ est également un ensemble convexe dans $\mathbb{R}^n$.*

Propriété 1.3. *Si E_1 est un ensemble convexe dans $\mathbb{R}^n$, alors*

$$E_2 = \{x^2 \in \mathbb{R}^m \ : \ x^2 = Ax^1 + b, \ x^1 \in E_1\},$$

où $A \in \mathbb{R}^{m \times n}$ et $b \in \mathbb{R}^m$, est un ensemble convexe dans $\mathbb{R}^m$.

1.5.3 Fonctions convexes

Soit E un ensemble convexe de $\mathbb{R}^n$, et soit $f : E \to \mathbb{R}$ une fonction réelle définie sur E.

Définition 1.12 (Fonction convexe)**.** La fonction f est dite convexe, si elle vérifie l'inégalité suivante pour tout $x, y \in E$ et tout $\lambda \in [0,1]$:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y). \tag{1.16}$$

De même, la fonction f est dite strictement convexe, si l'inégalité (1.16) est stricte pour tous $x, y \in E$, avec $x \neq y$ et $\lambda \in]0,1[$, i.e.

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y). \tag{1.17}$$

Définition 1.13 (Fonction concave)**.** La fonction f est dite concave, si pour tout $x, y \in E$ et tout $\lambda \in [0,1]$, on a :

$$f(\lambda x + (1-\lambda)y) \geq \lambda f(x) + (1-\lambda) f(y). \tag{1.18}$$

Autrement dit, f est concave si et seulement si $-f$ est convexe.

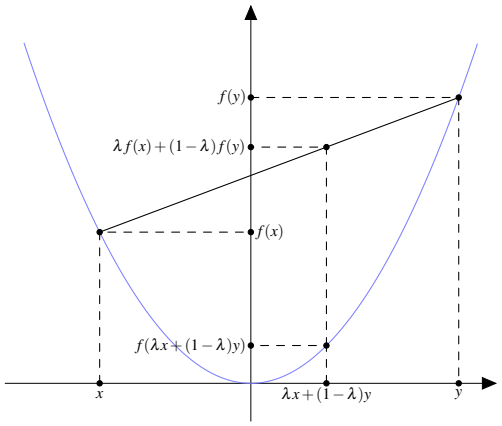

FIGURE 1.3 – Fonction convexe

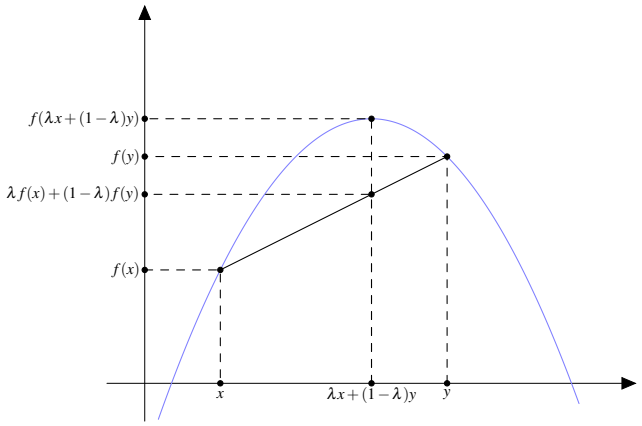

FIGURE 1.4 – Fonction concave

1.5.4 Hyperplan, polyèdre et polytope

Définition 1.14 (Hyperplan)**.** Un sous-ensemble H de $\mathbb{R}^n$ est un hyperplan s'il existe un vecteur non nul $a \in \mathbb{R}^n$, et un scalaire $\beta \in \mathbb{R}$, tels que

$$H = \{x \in \mathbb{R}^n \ : \ a^T x = \beta\}. \tag{1.19}$$

Cet hyperplan divise l'espace $\mathbb{R}^n$ en deux demi-espaces :

$$H^+ = \{x \in \mathbb{R}^n \ : \ a^T x \geq \beta\} \quad \text{et} \quad H^- = \{x \in \mathbb{R}^n \ : \ a^T x \leq \beta\}, \tag{1.20}$$

où H^+ et H^- sont convexes et fermés.

Définition 1.15 (Polyèdre)**.** Un sous-ensemble P de $\mathbb{R}^n$ est un polyèdre s'il est défini comme l'intersection d'un nombre fini de demi-espaces. Il est représenté sous la forme :

$$P = \{x \in \mathbb{R}^n \ : \ Ax \leq b\}, \tag{1.21}$$

où $A \in \mathscr{M}_{m\times n}(\mathbb{R})$ et b est un vecteur-colonne dans $\mathbb{R}^m$.

Remarque 1.4. Un polyèdre borné est appelé polytope.

Chapitre 2

Programmation linéaire et la programmation linéaire en nombres entiers

Sommaire

2.1 Introduction

L'objectif de ce chapitre est de présenter les fondements théoriques et algorithmiques de la Programmation Linéaire (PL) et de la Programmation Linéaire en Nombres Entiers (PLNE), deux approches essentielles en optimisation.

2.2 Programmation linéaire

2.2.1 Présentation du problème et définitions

La Programmation Linéaire (notée PL) constitue un puissant outil de la recherche opérationnelle, utilisé pour résoudre des problèmes d'optimisation où l'objectif est de minimiser ou maximiser une fonction linéaire (appelée fonction objectif) sous des contraintes linéaires. Ces problèmes se présentent dans divers domaines tels que la finance, le logistique, l'économie, etc.

2.2.1.1 Forme canonique

Le problème de PL se présente sous la forme suivante :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax \leq b, \\ & x \geq 0, \end{cases} \tag{2.1}$$

où $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathscr{M}_{m,n}(\mathbb{R})$.

Lorsque les variables de décision x sont bornées, il est appelé un problème de PL à variables bornées (PLvb) :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax \leq b, \\ & l \leq x \leq u, \end{cases} \tag{2.2}$$

où $l, u \in \mathbb{R}^n$, $l_j < u_j$, $j \in J = \{1, 2, \ldots, n\}$.

2.2.1.2 Forme standard

La forme standard implique la transformation des inégalités en des égalités en ajoutant des variables d'écart x^e :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax + I_m x^e = b, \\ & l \leq x \leq u, \quad l^e \leq x^e \leq u^e, \end{cases} \tag{2.3}$$

où

$$x^e = \begin{pmatrix} x^e_{n+1} \\ x^e_{n+2} \\ \vdots \\ x^e_{n+m} \end{pmatrix} \in \mathbb{R}^m, \quad l^e = \begin{pmatrix} l^e_{n+1} \\ l^e_{n+2} \\ \vdots \\ l^e_{n+m} \end{pmatrix} \in \mathbb{R}^m, \quad u^e = \begin{pmatrix} u^e_{n+1} \\ u^e_{n+2} \\ \vdots \\ u^e_{n+m} \end{pmatrix} \in \mathbb{R}^m,$$

et pour tout $i \in I$, on a :

$$\begin{cases} u^e_{n+i} = b_i - \sum\limits_{\substack{j\in J \\ a_{ij}>0}} a_{ij} l_j - \sum\limits_{\substack{j\in J \\ a_{ij}<0}} a_{ij} u_j, \\ l^e_{n+i} = 0. \end{cases} \tag{2.4}$$

En posant

$$y = (x, x^e)^T,\ \tilde{c} = (c, 0)^T,\ \tilde{l} = (l, l^e)^T,\ \tilde{u} = (u, u^e)^T,\ \tilde{A} = (A, I_m),$$

le problème peut être réécrit sous forme standard comme suit :

$$\begin{cases} \max & z(y) = \tilde{c}^T y, \\ & \tilde{A}\, y = b, \\ & \tilde{l} \le y \le \tilde{u}. \end{cases} \tag{2.5}$$

2.2.1.3 Notations

Nous définissons les ensembles d'indices suivants :

$$I = \{1, 2, \ldots, m\}, \quad J = \{1, 2, \ldots, n\},$$

$$J = J_B \cup J_N, \quad J_B \cap J_N = \emptyset, \quad |J_B| = |I| = m, \quad |J_N| = n - m.$$

Les vecteurs et la matrice A peuvent être écrits et partitionnés comme suit :

$$l = l(J) = (l_j,\ j \in J), \quad u = u(J) = (u_j,\ j \in J),$$

$$x = x(J) = (x_j,\ j \in J), \quad x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}, \quad x_B = x(J_B) = (x_j,\ j \in J_B), \quad x_N = x(J_N) = (x_j,\ j \in J_N),$$

$$c = c(J) = (c_j,\ j \in J), \quad c = \begin{pmatrix} c_B \\ c_N \end{pmatrix}, \quad c_B = c(J_B) = (c_j,\ j \in J_B), \quad c_N = c(J_N) = (c_j,\ j \in J_N),$$

$$A = A(I,\ J) = (a_{ij},\ i \in I,\ j \in J), \quad A = (A_B,\ A_N), \quad A_B = A(I,\ J_B), \quad A_N = A(I, J_N).$$

Définition 2.1 (Solution réalisable)**.** Un vecteur x qui satisfait les contraintes du problème (2.2) est une solution réalisable du problème de PLvb, où l'ensemble de ces solutions réalisables est noté S :

$$S = \{x \in \mathbb{R}^n\ :\ A\, x \le b,\ l \le x \le u\}. \tag{2.6}$$

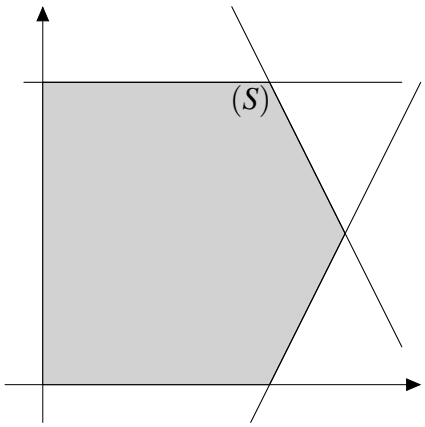

FIGURE 2.1 – Ensemble de solutions réalisables du problème (PL)

Définition 2.2 (Solution optimale)**.** Une solution réalisable x^0 du problème (PLvb) est optimale si elle réalise le maximum de la fonction objectif, c'est-à-dire :

$$z(x) \leq z(x^0), \quad \forall x \in S, \tag{2.7}$$

et on note :

$$z(x^0) = \max_{x \in S} z(x). \tag{2.8}$$

Définition 2.3 (Solution ε-optimale)**.** Une solution réalisable x^ε du problème (PLvb) est ε-optimale ou suboptimale si :

$$z(x^0) - z(x^\varepsilon) \leq \varepsilon, \tag{2.9}$$

où ε est un nombre positif ou nul choisi à l'avance, et x^0 étant une solution optimale de (PLvb).

2.2.2 Méthodes de résolution

Avant 1947, certaines méthodes permettaient de résoudre des cas particuliers de programmation linéaire, mais aucune ne traitait le problème général de manière systématique. En s'appuyant sur les travaux de J.B.J. Fourier [31] en 1826, G.B. Dantzig [26] a introduit en 1947 la méthode du simplexe, devenue une référence en raison de son efficacité sur les problèmes pratiques. Cependant, en 1972, Klee et Minty [49] ont construit un exemple montrant que cette méthode pouvait, dans le pire des cas, nécessiter un temps de calcul exponentiel.

2.2.2.1 Méthode adaptée

Dans cette partie, nous considérons le problème de PL sous la forme suivante :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax = b, \\ & l \leq x \leq u, \end{cases} \tag{2.10}$$

avec $rang(A) = m < n$.

Définition 2.4 (Support)**.** L'ensemble J_B est appelé support si :

$$\det(A_B) \neq 0. \tag{2.11}$$

Le couple $\{x,\ J_B\}$ formé de la solution réalisable x et du support J_B est appelé Solution Réalisable de Support (SRS).

Définition 2.5 (Solution non dégénérée)**.** Une solution réalisable de support $\{x,\ J_B\}$ est dite non dégénérée si chaque variable x_j pour $j \in J_B$ est strictement comprise entre ses bornes. Autrement dit,

$$l_j < x_j < u_j, \quad \forall j \in J_B. \tag{2.12}$$

Définition 2.6 (Solution réalisable de base)**.** Une Solution Réalisable de Base (SRB) est une solution réalisable de support $\{x,\ J_B\}$, où les variables x_j pour $j \in J_N$ prennent des valeurs critiques, c'est-à-dire

$$x_j = l_j \vee u_j, \quad \forall j \in J_N. \tag{2.13}$$

2.2.2.1.1 Formule d'accroissement de la fonction objectif

Soit $\{x,\ J_B\}$ une SRS du problème (2.10). Considérons une autre solution réalisable quelconque $\bar{x} = x + \Delta x$. L'accroissement de la fonction objectif s'écrit

$$\Delta z = z(\bar{x}) - z(x) = c^T \bar{x} - c^T x = c^T (x + \Delta x) - c^T x = c^T \Delta x.$$

En outre, les contraintes nous permettent d'écrire

$$A\ \bar{x} = b \Leftrightarrow A(x + \Delta x) = Ax + A\Delta x = b \Leftrightarrow A\ \Delta x = 0 \Leftrightarrow A_B \Delta x_B + A_N \Delta x_N = 0,$$

où

$$\Delta x = \begin{pmatrix} \Delta x_B \\ \Delta x_N \end{pmatrix},$$

donc

$$\Delta x_B = -A_B^{-1} A_N \Delta x_N.$$

D'après cette dernière relation, l'accroissement devient :

$$\begin{aligned} \Delta z = c^T \Delta x &= c_B^T \Delta x_B + c_N^T \Delta x_N \\ &= -c_B^T A_B^{-1} A_N \Delta x_N + c_N^T \Delta x_N \\ &= -\left(c_B^T A_B^{-1} A_N - c_N^T\right) \Delta x_N. \end{aligned}$$

Le vecteur des multiplicateurs π est défini comme suit :

$$\pi^T = c_B^T A_B^{-1}, \tag{2.14}$$

de même que le vecteur des coûts réduits E :

$$E^T = \pi^T A - c^T \Leftrightarrow E_j = \pi^T a_j - c_j, \quad j \in J, \tag{2.15}$$

où

$$E_B^T = \pi^T A_B - c_B^T = c_B^T A_B^{-1} A_B - c_B^T = 0, \quad \text{et} \quad E_N^T = \pi^T A_N - c_N^T.$$

Par conséquent, l'accroissement prend la forme finale suivante :

$$\Delta z = z(\bar{x}) - z(x) = -E_N^T \Delta x_N = -\sum_{j \in J_N} E_j \Delta x_j = -\sum_{j \in J_N} E_j (\bar{x}_j - x_j). \tag{2.16}$$

2.2.2.1.2 Critère d'optimalité et de supoptimalité

Définition 2.7. Considérons une SRS $\{x,\ J_B\}$ du problème (2.10), et définissons les ensembles d'indices suivants :

$$J_N^+ = \{j \in J_N\ :\ E_j > 0\}, \quad J_N^- = \{j \in J_N\ :\ E_j < 0\}, \quad J_N^0 = \{j \in J_N\ :\ E_j = 0\},$$

où

$$J_N = J \setminus J_B = J_N^+ \cup J_N^- \cup J_N^0.$$

Théorème 2.1 (Critère d'optimalité)**.** *[32] Soit $\{x,\ J_B\}$ une SRS du problème* (2.10)*. Les relations*

$$\begin{cases} E_j \geq 0 & \text{pour} \quad x_j = l_j, \\ E_j \leq 0 & \text{pour} \quad x_j = u_j, \\ E_j = 0 & \text{pour} \quad l_j < x_j < u_j, \quad j \in J_N, \end{cases} \tag{2.17}$$

sont suffisantes pour l'optimalité de la solution réalisable x. Elles sont aussi nécessaires dans le cas où la SRS $\{x,\ J_B\}$ est non dégénérée.

Démonstration. **Suffisance.** Soit $\{x,\ J_B\}$ une SRS vérifiant (2.17). Pour toute solution réalisable $\bar{x}$ du problème (2.10), la formule d'accroissement (2.16) nous donne

$$z(\bar{x}) - z(x) = -\sum_{j \in J_N^+} E_j (\bar{x}_j - l_j) - \sum_{j \in J_N^-} E_j (\bar{x}_j - u_j).$$

Comme on a

$$l_j \leq \bar{x}_j \leq u_j \Rightarrow \begin{cases} \bar{x}_j - l_j \ \geq 0, \\ \bar{x}_j - u_j \leq 0, \end{cases}$$

on en déduit que

$$z(\bar{x}) - z(x) \leq 0 \Rightarrow z(\bar{x}) \leq z(x).$$

Par conséquent, le vecteur x est une solution optimale du problème (2.10).

Nécessité. Soit $\{x,\ J_B\}$ une SRS optimale non dégénérée du problème (2.10) et supposons que les relations (2.17) ne sont pas vérifiées, il existe alors un indice j_0 tel que

$$j_0 \in J_N^+ \text{ et } x_{j_0} > l_{j_0} \quad \text{ou bien} \quad j_0 \in J_N^- \text{ et } x_{j_0} < u_{j_0}.$$

Construisons alors une autre solution réalisable $\bar{x} = x + \theta d$, où θ est un nombre réel positif et $d = d(J) = (d_B\ ,\ d_N) = (d\,(J_B)\ ,\ d\,((J_N))$ un vecteur de direction vérifiant

$$\begin{cases} d_{j_0} = -\text{sign}(E_{j_0}), \\ d_j\ = 0, & j \in J_N \setminus \{j_0\}, \\ d_B\ = -A_B^{-1} A_N d_N. \end{cases}$$

On a donc

$$A_B d_B + A_N d_N = Ad = 0 \Rightarrow A\bar{x} = A\,(x + \theta d) = Ax + \theta Ad = b.$$

Le vecteur $\bar{x}$ vérifie donc les contraintes principales $A\bar{x} = b$. Pour que $\bar{x}$ soit une solution réalisable du problème (2.10), il doit en plus vérifier l'inégalité

$$l \leq \bar{x} \leq u \Leftrightarrow l \leq x + \theta d \leq u \Leftrightarrow l - x \leq \theta d \leq u - x,$$

soit en écrivant composante par composante :

$$\begin{array}{lcll} l_j - x_j & \leq & \theta d_j & \leq u_j - x_j, \quad j \in J_B, \\ l_{j_0} - x_{j_0} & \leq & -\theta sign(E_{j_0}) & \leq u_{j_0} - x_{j_0}. \end{array} \tag{2.18}$$

Puisque la SRS $\{x,\ J_B\}$ est non dégénérée, on a alors

$$l_j - x_j < 0 < u_j - x_j, \quad j \in J_B.$$

De plus, on a

$$\begin{array}{ll} l_{j_0} - x_{j_0} < 0, & \text{si} \quad E_{j_0} > 0, \\ u_{j_0} - x_{j_0} > 0, & \text{si} \quad E_{j_0} < 0. \end{array}$$

Alors pour un nombre θ strictement positif assez petit, les relations (2.18) seront vérifiées et le vecteur $\bar{x}$ sera une solution réalisable du problème (2.10). La formule d'accroissemnt (2.16) nous donne donc

$$\begin{aligned} z(\bar{x}) - z(x) = -\sum_{j \in J_N} E_j(\bar{x}_j - x_j) = -\theta \sum_{j \in J_N} E_j d_j = -\theta E_{j_0} d_{j_0}, \\ = \theta E_{j_0} \text{sign}(E_{j_0}) = \theta |E_{j_0}| > 0. \end{aligned} \tag{2.19}$$

On en déduit que $z(\bar{x}) > z(x)$. Mais ceci est en contradiction avec le fait que x est une solution optimale du problème (2.10). Par conséquent, si $\{x,\ J_B\}$ est une SRS optimale non dégénérée, alors les relations (2.17) sont forcément vérifiées.

□

Pour estimer l'écart qui existe entre la valeur optimale $z(x^0)$ et une autre valeur $z(x)$ d'une SRS quelconque $\{x,\ J_B\}$, remplaçons dans la formule d'accroissement (2.16) le vecteur $\bar{x}$ par x^0 et majorons la valeur de l'expression (2.16). On aura donc

$$z(x^0)-z(x)=-\sum_{j\in J_N} E_j(x_j^0-x_j)=\sum_{j\in J_N^+} E_j(x_j-x_j^0)+\sum_{j\in J_N^-} E_j(x_j-x_j^0). \tag{2.20}$$

Puisque la solution optimale x^0 vérifie $l_j\le x_j^0\le u_j$, $j\in J$, alors il en résulte que

$$\begin{cases} x_j-x_j^0\le x_j-l_j,\\ x_j-x_j^0\ge x_j-u_j. \end{cases}$$

Donc

$$\begin{cases} E_j(x_j-x_j^0)\le E_j(x_j-l_j), & \text{si} \quad E_j>0,\\ E_j(x_j-x_j^0)\le E_j(x_j-u_j), & \text{si} \quad E_j<0. \end{cases}$$

Par conséquent, on obtient une majoration du membre inconnu de droite de l'égalité (2.20) :

$$z(x^0)-z(x)\le \sum_{j\in J_N^+} E_j(x_j-l_j)+\sum_{j\in J_N^-} E_j(x_j-u_j). \tag{2.21}$$

Le nombre

$$\beta(x,J_B)=\sum_{j\in J_N^+} E_j(x_j-l_j)+\sum_{j\in J_N^-} E_j(x_j-u_j) \tag{2.22}$$

est ainsi appelé estimation de suboptimalité.

Théorème 2.2 (condition suffisante de suboptimalité)**.** *Soit* $\{x,\ J_B\}$ *une SRS du problème* (2.10) *et* ε *un nombre positif ou nul arbitraire. Si*

$$\beta(x,J_B)\le\varepsilon, \tag{2.23}$$

alors la solution réalisable x est ε*-optimale.*

Démonstration. En vertu de (2.21) et (2.22), on peut écrire

$$z(x^0)-z(x)\le\beta(x,J_B)\le\varepsilon\Rightarrow z(x^0)-z(x)\le\varepsilon.$$

La solution réalisable x est donc ε-optimale. □

2.2.2.1.3 Une itération de l'algorithme

Étant donné un nombre réel positif ou nul quelconque ε et une SRS initiale $\{x,\ J_B\}$, l'objectif de l'algorithme est de construire une solution réalisable ε-optimale x^ε ou idéalement, une solution optimale x^0. L'itération de l'algorithme consiste à passer de $\{x,\ J_B\}$ à $\{\bar{x},\bar{J}_B\}$ telle que $z(\bar{x})\ge z(x)$. Pour ce faire, on construit la nouvelle solution réalisable $\bar{x}$ de la manière suivante :

$$\bar{x} = x + \theta^0 d, \quad \theta^0 \geq 0,$$

où le vecteur $d \in \mathbb{R}^n$, est appelé direction adaptée d'amélioration, construit comme suit :

$$\begin{cases} d_j = l_j - x_j, & \text{si} \quad j \in J_N^+, \\ d_j = u_j - x_j, & \text{si} \quad j \in J_N^-, \\ d_j = 0, & \text{si} \quad j \in J_N^0, \\ d_B = -A_B^{-1} A_N d_N, \end{cases} \tag{2.24}$$

et θ^0 est le pas le long cette direction, devant satisfaire les relations suivantes :

$$l_j - x_j \leq \quad \theta^0 d_j \quad \leq u_j - x_j, \quad j \in J_N, \tag{2.25}$$

$$l_j - x_j \leq \quad \theta^0 d_j \quad \leq u_j - x_j, \quad j \in J_B. \tag{2.26}$$

D'après les inégalités (2.25), et les formules (2.24) concerant les indices de J_N, on conclut que $\theta^0 \leq 1$. Pour vérifier les inégalités (2.26), il faut choisir θ^0 tel que $\theta^0 \leq \theta_{j_1}$, où

$$\theta_{j_1} = \min_{j \in J_B} \theta_j, \quad \text{avec} \quad \theta_j = \begin{cases} \dfrac{u_j - x_j}{d_j}, & \text{si} \quad d_j > 0, \\ \dfrac{l_j - x_j}{d_j}, & \text{si} \quad d_j < 0, \\ \infty, & \text{si} \quad d_j = 0. \end{cases} \tag{2.27}$$

Donc :

$$\theta^0 = \min\{\theta_{j_1}, 1\}, \quad \bar{x} = x + \theta^0 d. \tag{2.28}$$

Il y a alors deux cas possibles qui peuvent se présenter :

(i) $\theta^0 = 1$: La SRS $\bar{x} = x + d$ est optimale, car elle vérifie le critère d'optimalité. Par conséquent, le processus de résolution du problème (2.10) est terminé.

(ii) $\theta^0 = \theta_{j_1} < 1$: Si tel est le cas, la valeur de la fonction objectif augmentera d'une quantité égale à :

$$z(\bar{x}) - z(x) = -\theta^0 \sum_{j \in J_N} E_j d_j = \theta^0 \beta(x, J_B). \tag{2.29}$$

De plus, l'estimation de suboptimalité de la nouvelle SRS $\{\bar{x}, J_B\}$ sera calculée comme suit :

$$\begin{aligned} \beta(\bar{x}, J_B) &= \sum_{j \in J_N^+} E_j(\bar{x}_j - l_j) + \sum_{j \in J_N^-} E_j(\bar{x}_j - u_j) \\ &= \sum_{j \in J_N^+} E_j(x_j + \theta^0 d_j - l_j) + \sum_{j \in J_N^-} E_j(x_j + \theta^0 d_j - u_j) \\ &= \beta(x, J_B) + \theta^0 \sum_{j \in J_N^+} E_j d_j + \theta^0 \sum_{j \in J_N^-} E_j d_j \\ &= \beta(x, J_B) - \theta^0 \beta(x, J_B) \\ &= (1 - \theta^0)\beta(x, J_B). \end{aligned} \tag{2.30}$$

Si $\beta(\bar{x}, J_B\} \leq \varepsilon$, la SRS $\bar{x}$ est ε-optimale et le processus de résolution du problème (2.10) s'arrête. Sinon, le support J_B doit être changé, car la composante $\bar{x}_{j_1}$ atteint une valeur critique et le vecteur a_{j_1} doit sortir de la base A_B. Pour cela, on cherche à remplacer a_{j_1} par un vecteur a_{j_0} avec $j_0 \in J_N$, de sorte que l'ensemble $\overline{J}_B = (J_B \setminus \{j_1\}) \cup \{j_0\}$ soit toujours un support. Pour cela, on définit le vecteur $\kappa \in \mathbb{R}^n$, appelé pseudo-solution, et le nombre réel α_0 comme suit :

$$\kappa = x + d \quad \text{et} \quad \alpha_0 = \kappa_{j_1} - \bar{x}_{j_1}. \tag{2.31}$$

La direction duale $t = (t_j,\ j \in J)$ est obtenue à partir de :

$$\begin{cases} t_{j_1} = -\text{sign}(\alpha_0), \\ t_j = 0, & j \in J_B \setminus \{j_1\}, \\ t_N^T = t_B^T A_B^{-1} A_N. \end{cases} \tag{2.32}$$

Le calcul du pas dual σ^0 est déterminé par

$$\sigma^0 = \sigma_{j_0} = \min_{j \in J_N} \sigma_j, \tag{2.33}$$

où les σ_j sont calculés à l'aide des relations suivantes :

$$\sigma_j = \begin{cases} \dfrac{-E_j}{t_j} & \text{si} \quad E_j t_j < 0, \\ 0 & \text{si} \quad E_j = 0, \quad t_j < 0, \quad \kappa_j \neq u_j, \\ 0 & \text{si} \quad E_j = 0, \quad t_j > 0, \quad \kappa_j \neq l_j, \\ +\infty & \text{dans les autres cas.} \end{cases} \tag{2.34}$$

Le nouveau vecteur des coûts réduits, et le nouveau support sont déterminés comme suit :

$$\overline{E} = E + \sigma^0 t, \quad \overline{J}_B = (J_B \setminus \{j_1\}) \cup \{j_0\}. \tag{2.35}$$

L'estimation de suboptimalité correspondant à la nouvelle solution et au nouveau support est :

$$\beta(\bar{x}, \overline{J}_B) = (1 - \theta^0)\beta(x, J_B) - \sigma^0 |\alpha_0| = \beta(\bar{x}, J_B) - \sigma^0 |\alpha_0|. \tag{2.36}$$

Si $\beta(\bar{x}, \overline{J}_B) > \varepsilon$, alors on recommence une nouvelle itération en posant :

$$x := \bar{x}, \quad J_B := \overline{J}_B, \quad E := \overline{E}, \quad \beta(x, J_B) := \beta(\bar{x}, \overline{J}_B).$$

Sinon, le processus de résolution du problème (2.10) s'arrête.

2.2.2.1.4 Schéma de l'algorithme

Algorithme 1 : Méthode Adaptée avec la règle du pas simple

Entrées : A, b, c, J_B, J_N, x, ε
Sorties : x^0, z^0

Calculer π, E et $z = c^T x$
Calculer $\beta(x, J_B)$
tant que $\beta(x, J_B) > \varepsilon$ **faire**
  Calculer la direction d'amélioration d et le pas θ^0
  Mettre à jour $\overline{x}$ et $\overline{z}$
  si $\theta^0 = 1$ **alors**
    Poser $x := \overline{x}$, $z := \overline{z}$ et $\beta(x, J_B) = 0$
  sinon
    Calculer $\beta(\overline{x}, J_B)$
    si $\beta(\overline{x}, J_B) \leq \varepsilon$ **alors**
      Poser $x := \overline{x}$, $z := \overline{z}$ et $\beta(x, J_B) := \beta(\overline{x}, J_B)$
    sinon
      Calculer la pseudo-solution κ et α_0
      Calculer la direction duale t et le pas dual σ^0
      Mettre à jour $\overline{E}$, $\overline{J}_B$ et $\beta(\overline{x}, \overline{J}_B)$
      Poser $x := \overline{x}$, $z := \overline{z}$, $J_B := \overline{J}_B$, $E := \overline{E}$ et $\beta(x, J_B) := \beta(\overline{x}, \overline{J}_B)$
    fin si
  fin si
fin tq
si $\beta(x, J_B) = 0$ **alors**
  $\{x, J_B\}$ une SRS optimale
sinon
  $\{x, J_B\}$ une SRS ε-optimale
fin si
La solution optimale (ou ε-optimale) est : $x^0 = x$
La valeur maximale de la fonction objectif est : $z^0 = z$
retourne x^0, z^0 ;

Exemple 2.1. Considérons le problème de programmation linéaire à variables bornées :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax \leq b, \\ & l \leq x \leq u, \end{cases}$$

où

$$A = \begin{pmatrix} 1 & -1 & 3 & 2 \\ -5 & 1 & 2 & 3 \end{pmatrix}, \quad b = \begin{pmatrix} 8 \\ 2 \end{pmatrix}, \quad c = \begin{pmatrix} 2 \\ -3 \\ 1 \\ 1 \end{pmatrix}, \quad l = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad u = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}.$$

La forme standard s'écrit

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax + I_2 x^e = b, \\ & l \leq x \leq u, \quad l^e \leq x^e \leq u^e, \end{cases}$$

avec

$$x^e = \begin{pmatrix} x_5 \\ x_6 \end{pmatrix}, \quad l^e = \begin{pmatrix} l_5 \\ l_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

$$\begin{cases} u_5 = & b_1 - a_{11}l_1 - a_{21}u_2 - a_{31}l_3 - a_{41}l_4 = 10, \\ u_6 = & b_2 - a_{12}u_1 - a_{22}l_2 - a_{32}l_3 - a_{42}l_4 = 7, \end{cases} \Leftrightarrow u^e = \begin{pmatrix} u_5 \\ u_6 \end{pmatrix} = \begin{pmatrix} 10 \\ 7 \end{pmatrix}$$

On pose

$$y = (x, x^e)^T, \quad \tilde{A} = (A|I_2), \quad \tilde{c} = (c, 0_{\mathbb{R}^2})^T, \quad \tilde{l} = (l, l^e)^T, \quad \tilde{u} = (u, u^e)^T.$$

La SRS initiale $\{y,\ J_B\}$ pour ce problème est :

$$y = (0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 8\ ,\ 2)^T, \quad z = z(y) = 0, \quad J_B = \{5,6\}, \quad J_N = \{1,2,3,4\}, \quad \varepsilon = 10^{-3}.$$

Itération 1.

$$\tilde{A}_B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \tilde{A}_N = \begin{pmatrix} 1 & -1 & 3 & 2 \\ -5 & 1 & 2 & 3 \end{pmatrix}, \quad \tilde{c}_B = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \tilde{c}_N = \begin{pmatrix} 2 \\ -3 \\ 1 \\ 1 \end{pmatrix}.$$

— Calcul du vecteur des multiplicateurs π et du vecteur des coûts réduits E :

$$\pi^T = \tilde{c}_B^T \tilde{A}_B^{-1} = (0\ ,\ 0), \quad E^T = \pi^T \tilde{A} - \tilde{c}^T = (-2\ ,\ 3\ ,\ -1\ ,\ -1\ ,\ 0\ ,\ 0).$$

— L'estimation de suboptimalité $\beta(y, J_B)$:

$$\beta(y, J_B) = E_1(x_1 - u_1) + E_2(x_2 - l_2) + E_3(x_3 - u_3) + E_4(x_4 - u_4) = 9 > \varepsilon.$$

Donc y n'est pas optimale.

— Calcul de la direction d'amélioration d :

$$\begin{cases} d_1 = u_1 - x_1 = 1, \\ d_2 = l_2 - x_2 = 0, \\ d_3 = u_3 - x_3 = 3, \\ d_4 = u_4 - x_4 = 4, \end{cases} \Rightarrow d_N = (d_1\ ,\ d_2\ ,\ d_3\ ,\ d_4)^T = (1\ ,\ 0\ ,\ 3\ ,\ 4)^T.$$

$$d_B = (d_5\ ,\ d_6)^T = -\tilde{A}_B^{-1} \tilde{A}_N d_N = (-18\ ,\ -13)^T \Rightarrow d = (1\ ,\ 0\ ,\ 3\ ,\ 4\ ,\ -18\ ,\ -13)^T.$$

— Calcul du pas θ^0 le long de la direction d :

$$\begin{cases} \theta_5 = \dfrac{l_5 - x_5}{d_5} = \dfrac{4}{9}, \\ \theta_6 = \dfrac{l_6 - x_6}{d_6} = \dfrac{2}{13}, \end{cases} \Rightarrow \theta^0 = \min\{1, \theta_5, \theta_6\} = \theta_6 = \frac{2}{13} \Rightarrow j_1 = 6.$$

— Calcul de la nouvelle solution réalisable $\overline{y}$ et $\overline{z}$:

$$\overline{y} = y + \theta^0 d = \left(\frac{2}{13}\ ,\ 0\ ,\ \frac{6}{13}\ ,\ \frac{8}{13}\ ,\ \frac{68}{13}\ ,\ 0\right)^T, \quad \overline{z} = z + \theta^0 \beta(y, J_B) = \frac{18}{13} \approx 1.38.$$

— Calcul de l'estimation de suboptimalité $\beta(\overline{y}, J_B)$:

$$\beta(\overline{y}, J_B) = (1 - \theta^0)\beta(y, J_B) = \frac{99}{13} \approx 7.61 > \varepsilon,$$

donc $\overline{y}$ n'est pas optimale.

— Calcul du vecteur κ et de la direction duale t :

$$\kappa = y + d = (1\ ,\ 0\ ,\ 3\ ,\ 4\ ,\ -10\ ,\ -11)^T, \quad \alpha_0 = \kappa_6 - \overline{x}_6 = -11,$$

$$\begin{cases} t_6 = -\text{sign}(\alpha_0) = 1, \\ t_5 = 0 \end{cases} \Rightarrow t_B = (t_5\ ,\ t_6)^T = (0\ ,\ 1)^T,$$

$$t_N = (t_1\ ,\ t_2\ ,\ t_3\ ,\ t_4)^T = t_B^T \tilde{A}_B^{-1} \tilde{A}_N = (-5\ ,\ 1\ ,\ 2\ ,\ 3)^T \Rightarrow t = (-5\ ,\ 1\ ,\ 2\ ,\ 3\ ,\ 0\ ,\ 1)^T.$$

— Calcul du pas dual et de l'indice j_0 :

$$\begin{cases} \sigma_1 = \sigma_2 = +\infty, \\ \sigma_3 = \dfrac{1}{2}, \\ \sigma_4 = \dfrac{1}{3}, \end{cases} \Rightarrow \sigma^0 = \min\{\sigma_3, \sigma_4\} = \min\left\{\frac{1}{2}, \frac{1}{3}\right\} = \sigma_4 = \frac{1}{3} \Rightarrow j_0 = 4.$$

— Le nouveau support et le nouveau vecteur des coûts réduits sont donc égaux à :

$$\overline{J}_B = (J_B \setminus \{j_1\}) \cup \{j_0\} = \{5, 4\}, \quad \overline{E} = E + \sigma^0 t = \left(-\frac{11}{3}, \frac{10}{3}, -\frac{1}{3}, 0, 0, \frac{1}{3}\right)^T.$$

— L'estimation de suboptimalité vaut donc :

$$\beta(\overline{y}, \overline{J}_B) = \beta(\overline{y}, J_B) - \sigma^0 |\alpha_0| = \frac{154}{39} \approx 3.94.$$

Itération 2. On a :

$$y = \left(\frac{2}{13}\ ,\ 0\ ,\ \frac{6}{13}\ ,\ \frac{8}{13}\ ,\ \frac{68}{13}\ ,\ 0\right)^T, \quad z = \frac{18}{13}, \quad J_B = \{5, 4\}, \quad J_N = \{1, 2, 3, 6\}, \quad \beta(y, J_B) = \frac{154}{39},$$

$$E = \left(-\frac{11}{3}\ ,\ \frac{10}{3}\ ,\ -\frac{1}{3}\ ,\ 0\ ,\ 0\ ,\ \frac{1}{3}\right)^T, \quad \tilde{A}_B = \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix}, \quad \tilde{A}_N = \begin{pmatrix} 1 & -1 & 3 & 0 \\ -5 & 1 & 2 & 1 \end{pmatrix}.$$

— Calcul de la direction d'amélioration d :

$$\begin{cases} d_1 = u_1 - x_1 = \dfrac{11}{13}, \\ d_2 = l_2 - x_2 = 0, \\ d_3 = u_3 - x_3 = \dfrac{33}{13}, \\ d_6 = l_6 - x_6 = 0, \end{cases} \Rightarrow d_N = (d_1\ ,\ d_2\ ,\ d_3\ ,\ d_6)^T = \left(\frac{11}{13}\ ,\ 0\ ,\ \frac{33}{13}\ ,\ 0\right)^T,$$

$$d_B = (d_5\ ,\ d_4)^T = -\tilde{A}_B^{-1}\tilde{A}_N d_N = \left(-\frac{308}{39}\ ,\ -\frac{11}{39}\right)^T \Rightarrow d = \left(\frac{11}{13}\ ,\ 0\ ,\ \frac{33}{13}\ ,\ -\frac{11}{39}\ ,\ -\frac{308}{39}\ ,\ 0\right)^T.$$

— Calcul du pas θ^0 le long de la direction d :

$$\begin{cases} \theta_5 = \dfrac{l_5 - x_5}{d_5} = \dfrac{51}{77}, \\ \theta_4 = \dfrac{l_4 - x_4}{d_4} = \dfrac{24}{11}, \end{cases} \Rightarrow \theta^0 = \min\{1, \theta_5, \theta_4\} = \theta_5 = \frac{51}{77} \Rightarrow j_1 = 5.$$

— Calcul de la nouvelle solution réalisable $\overline{y}$ et $\overline{z}$:

$$\overline{y} = y + \theta^0 d = \left(\frac{5}{7}\ ,\ 0\ ,\ \frac{15}{7}\ ,\ \frac{3}{7}\ ,\ 0\ ,\ 0\right)^T, \quad \overline{z} = z + \theta^0 \beta(y, J_B) = 4.$$

— Calcul de l'estimation de suboptimalité $\beta(\overline{y}, J_B)$:

$$\beta(\overline{y}, J_B) = (1 - \theta^0)\beta(y, J_B) = \frac{4}{3} \approx 1.33 > \varepsilon,$$

donc $\overline{y}$ n'est pas optimale.

— Calcul du vecteur κ et de la direction duale t :

$$\kappa = y + d = \left(1\ ,\ 0\ ,\ 3\ ,\ \frac{1}{3}\ ,\ -\frac{8}{3}\ ,\ 0\right)^T, \quad \alpha_0 = \kappa_5 - \overline{y}_5 = -\frac{8}{3},$$

$$\begin{cases} t_5 = -\text{sign}(\alpha_0) = 1, \\ t_4 = 0, \end{cases} \Rightarrow t_B = (t_5\ ,\ t_4)^T = (1\ ,\ 0)^T,$$

$$t_N = (t_1\ ,\ t_2\ ,\ t_3\ ,\ t_6)^T = t_B^T \tilde{A}_B^{-1}\tilde{A}_N = \left(\frac{13}{3}\ ,\ -\frac{5}{3}\ ,\ \frac{5}{3}\ ,\ -\frac{2}{3}\right)^T \Rightarrow t = \left(\frac{13}{3}\ ,\ -\frac{5}{3}\ ,\ \frac{5}{3}\ ,\ 0\ ,\ 1\ ,\ -\frac{2}{3}\right)^T.$$

— Calcul du pas dual et de l'indice j_0 :

$$\begin{cases} \sigma_1 = \dfrac{11}{13}, \\ \sigma_2 = 2, \\ \sigma_3 = \dfrac{1}{5}, \\ \sigma_6 = \dfrac{1}{2}, \end{cases} \Rightarrow \sigma^0 = \min\{\sigma_1, \sigma_2, \sigma_3, \sigma_6\} = \min\left\{\frac{11}{13}, 2, \frac{1}{5}, \frac{1}{2}\right\} = \sigma_3 = \frac{1}{5} \Rightarrow j_0 = 3.$$

— Le nouveau support et le nouveau vecteur des coûts réduits sont donc égaux à :

$$\overline{J}_B = (J_B \setminus \{j_1\}) \cup \{j_0\} = \{3,4\}, \quad \overline{E} = E + \sigma^0 t = \left(-\frac{14}{5}\ ,\ 3\ ,\ 0\ ,\ 0\ ,\ \frac{1}{5}\ ,\ \frac{1}{5}\right)^T.$$

— L'estimation de suboptimalité vaut donc :

$$\beta(\overline{y}, \overline{J}_B) = \beta(\overline{y}, J_B) - \sigma^0 |\alpha_0| = \frac{4}{5} \approx 0.8.$$

Itération 3. On a :

$$y = \left(\frac{5}{7}\ ,\ 0\ ,\ \frac{15}{7}\ ,\ \frac{3}{7}\ ,\ 0\ ,\ 0\right)^T, \quad z = 4, \quad J_B = \{3,4\}, \quad J_N = \{1,2,5,6\}, \quad \beta(y, J_B) = \frac{4}{5},$$

$$E = \left(-\frac{14}{5}\ ,\ 3\ ,\ 0\ ,\ 0\ ,\ \frac{1}{5}\ ,\ \frac{1}{5}\right)^T, \quad \tilde{A}_B = \begin{pmatrix} 3 & 2 \\ 2 & 3 \end{pmatrix}, \quad \tilde{A}_N = \begin{pmatrix} 1 & -1 & 1 & 0 \\ -5 & 1 & 0 & 1 \end{pmatrix}.$$

— Calcul de la direction d'amélioration d :

$$\begin{cases} d_1 = u_1 - x_1 = \dfrac{2}{7}, \\ d_2 = l_2 - x_2 = 0, \\ d_5 = l_5 - x_5 = 0, \\ d_6 = l_6 - x_6 = 0, \end{cases} \Rightarrow d_N = (d_1\ ,\ d_2\ ,\ d_5\ ,\ d_6)^T = \left(\frac{2}{7}\ ,\ 0\ ,\ 0\ ,\ 0\right)^T.$$

$$d_B = (d_3\ ,\ d_4)^T = -\tilde{A}_B^{-1} \tilde{A}_N d_N = \left(-\frac{26}{35}\ ,\ \frac{34}{35}\right)^T \Rightarrow d = \left(\frac{2}{7}\ ,\ 0\ ,\ -\frac{26}{35}\ ,\ \frac{34}{35}\ ,\ 0\ ,\ 0\right)^T.$$

— Calcul du pas θ^0 le long de la direction d :

$$\begin{cases} \theta_3 = \dfrac{l_3 - x_3}{d_3} = \dfrac{75}{26}, \\ \theta_4 = \dfrac{u_4 - x_4}{d_4} = \dfrac{125}{34}, \end{cases} \Rightarrow \theta^0 = \min\{1, \theta_3, \theta_4\} = 1.$$

— Calcul de la nouvelle solution réalisable $\overline{y}$ et $\overline{z}$:

$$\overline{y} = y + \theta^0 d = \left(1\ ,\ 0\ ,\ \frac{7}{5}\ ,\ \frac{7}{5}\ ,\ 0\ ,\ 0\right)^T, \quad \overline{z} = z + \theta^0 \beta(y, J_B) = \frac{24}{5} = 4.8.$$

Puisque $\theta^0 = 1$, le vecteur $x^0 = \left(1\ ,\ 0\ ,\ \frac{7}{5}\ ,\ \frac{7}{5}\right)^T$ est alors une solution optimale du problème considéré, avec $z^0 = z(x^0) = \frac{24}{5}$.

2.2.2.2 Méthode du simplexe

La méthode du simplexe est une méthode itérative utilisée pour résoudre des problèmes de programmation linéaire. Son principe fondamental repose sur la recherche d'une solution optimale en se déplaçant d'un sommet à un autre, dans l'espace des solutions réalisables S, avec une meilleure valeur de la fonction objective,

Algorithme 2 : Algorithme du simplexe

Entrées : *Le problème de PL*
Sorties : x^0, z^0

Convertir le problème de PL en sa forme standard
Calculer la solution de base réalisable initiale : $x_B = A_B^{-1} b$
Calculer π et E
tant que *E sont non-positifs* **faire**
  Sélectionner une variable non-basique entrante x_{j_0}, où $j_0 \in J_N$ est choisi selon $j_0 = \arg\min_{j \in J_N} \{E_j : E_j < 0\}$
  Calculer le direction de base $d_B = -A_B^{-1} a_{j_0}$
  si $d_i \geq 0$ *pour tout* $i \in J_B$ **alors**
    Le problème est non borné, Stop
    retourne NULL ;
  sinon
    Parmi les variables basiques actuelles, trouver la variable sortante x_{i_0}, où $i_0 \in J_B$ est choisi selon $i_0 = \arg\min_{i \in J_B} \left\{ -\frac{x_i}{d_i} : d_i < 0 \right\}$
    Calculer le pas $\theta^0 = -\frac{x_{i_0}}{d_{i_0}}$
    Calculer la nouvelle solution $x_{j_0} := \theta^0, \; x_B := x_B + \theta^0 d_B$
    Mettre à jour la base J_B et les coûts réduits E
  fin si
fin tq
La solution optimale est : $x_B^0 = x_B, x_N^0 = 0_{\mathbb{R}^{n-m}}$
La valeur maximale de la fonction objectif est : $z^0 = {c_B}^T x_B^0$
retourne x^0, z^0 ;

2.3 Programmation linéaire en nombres entiers

La Programmation Linéaire en Nombres Entiers (PLNE) est un domaine de la recherche opérationnelle qui vise à trouver des solutions optimales à des problèmes d'optimisation linéaire, où toutes les variables doivent être des nombres entiers. Cette restriction peut rendre la résolution du problème plus complexe, cependant cette restriction s'avère fondamentale, car elle permet de modéliser de manière plus précise des situations réelles qui impliquent des choix discrets. La PLNE est utilisée dans de nombreux domaines, tels que la logistique et le transport, la planification de la production et la finance.

2.3.1 Formulations mathématiques

Considérons un problème de PLNE écrit sous la forme suivante :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax \leq b, \\ & l \leq x \leq u, \quad x \in \mathbb{Z}^n, \end{cases} \tag{2.37}$$

où $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ et $c,l,u \in \mathbb{R}^n$, et $x \in \mathbb{Z}^n$ indique que les variables de décision x_j sont des entiers relatifs.

Dans un problème de PLNE, si les variables x_j sont restreintes à prendre uniquement des valeurs 0 ou 1, alors ce type de problème est appelé Programmation Linéaire en Nombres Binaires (PLNB). Il peut être formulé comme suit :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax \leq b, \\ & x \in \{0,1\}^n. \end{cases} \tag{2.38}$$

Une étape importante dans la résolution des problèmes de PLNE est la relaxation linéaire. En relaxant les contraintes d'intégralité sur les variables, nous pouvons formuler le problème relaxé (PR) associé sous forme de PL :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax \leq b, \\ & l \leq x \leq u, \quad x \in \mathbb{R}^n. \end{cases} \tag{2.39}$$

La relaxation linéaire vise à simplifier la résolution du problème, et sa solution optimale constitue une borne supérieure de la solution du problème de PLNE.

Remarque 2.1. Le problème (2.39) est une relaxation du problème (2.37), car l'ensemble des solutions réalisables du problème de PLNE est inclus dans l'ensemble des solutions réalisables du PR.

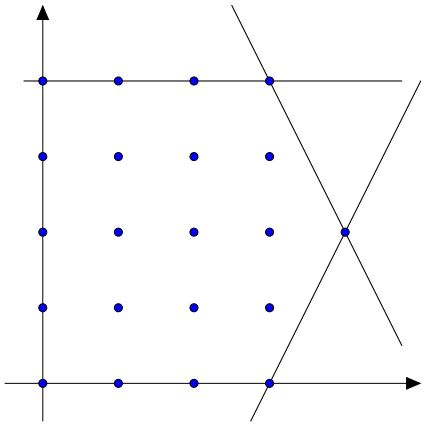

FIGURE 2.2 – Ensemble de solutions réalisables du problème (PLNE)

Corollaire 2.1. *Si la solution optimale du PR est entière, alors elle est également optimale pour le problème de PLNE.*

2.3.2 Méthodes de résolution

Différentes méthodes existent pour résoudre les PLNE, et elles peuvent être classées en deux catégories principales :

Méthodes exactes : Ces méthodes garantissent de trouver la solution optimale du problème, mais peuvent être coûteuses en temps de calcul pour les problèmes de grande taille. Parmi les méthodes exactes les plus connues, on retrouve Branch and Bound et les méthodes de coupes (comme les coupes de Gomory).

Méthodes heuristiques : Ces méthodes ne garantissent pas de trouver une solution optimale, mais peuvent fournir de bonnes solutions approchées en un temps de calcul raisonnable, notamment pour les problèmes de grande taille. Parmi les méthodes heuristiques, on peut citer celle de l'arrondi.

2.3.2.1 Méthode de Branch and Bound

La méthode de Branch and Bound (B&B) est l'une des méthodes les plus classiques pour résoudre des problèmes de PLNE. Son principe fondamental consiste à décomposer le problème initial en une arborescence de sous-problèmes, puis à explorer l'arbre de recherche de manière systématique.

La phase de séparation intervient lorsqu'une solution partielle n'est pas entière. On choisit une variable non entière x_j^0 telle que :

$$\lfloor x_j^0 \rfloor < x_j^0 < \lceil x_j^0 \rceil, \tag{2.40}$$

où $\lfloor x_j^0 \rfloor$ est la partie entière inférieure de x_j^0 et $\lceil x_j^0 \rceil$ est la partie entière supérieure de x_j^0. On divise ensuite le problème (P_k) en deux sous-problèmes (P_{k_1}) et (P_{k_2}) :

$$(\mathrm{P}_{k_1}) \begin{cases} (\mathrm{P}_k), \\ x_j \leq \lfloor x_j^0 \rfloor, \end{cases} \qquad (\mathrm{P}_{k_2}) \begin{cases} (\mathrm{P}_k), \\ x_j \geq \lceil x_j^0 \rceil. \end{cases}$$

La phase d'évaluation consiste à résoudre chaque sous-problème. Cela implique souvent la résolution d'un problème relaxé pour obtenir des bornes supérieures. Ces bornes sont importantes pour élaguer les branches non prometteuses.

Algorithme 3 : Algorithme de Branch and Bound

Entrées : *Le problème de PLNE*
Sorties : *Solution optimale x^0, valeur maximale z^0*

Relaxer le problème de PLNE pour obtenir un problème relaxé (P_0)
Initialiser une liste de sous-problèmes $\Gamma = \{(P_0)\}$, $x^* = \emptyset$ et $z^* = -\infty$
tant que $\Gamma \neq \emptyset$ **faire**
  Sélectionner et retirer un sous-problème (P_k) de Γ
  si *(P_k) est admissible* **alors**
    Résoudre le problème (P_k) pour obtenir x^k et z^k
    si $x^k \in \mathbb{Z}$ *et* $z^k > z^*$ **alors**
      Mettre à jour la meilleure solution entière $x^* := x^k$ et $z^* := z^k$
      Supprimer de Γ tous les sous-problèmes (P_j) tels que $z^j \leq z^0$
    fin si
    si $x^* \notin \mathbb{Z}$ *et* $z^k > z^*$ **alors**
      Diviser (P_k) en deux sous-problèmes (P_{k_1}) et (P_{k_2}) en ajoutant des contraintes supplémentaires
      Ajouter les nouveaux sous-problèmes à Γ
    fin si
  fin si
fin tq
La solution optimale est : $x^0 = x^*$
La valeur maximale de la fonction objectif est : $z^0 = z^*$
retourne x^0, z^0 ;

2.3.2.2 Méthode des coupes de Gomory

La méthode des coupes de Gomory est une technique de résoulution exacte pour les problèmes de PLNE. Elle est basée sur l'ajout itératif des contraintes linéaires, appelées coupes de Gomory, au problème relaxé dont la solution optimale est fractionnaire. Ces coupes éliminent des solutions fractionnaires sans exclure aucune solution entière réalisable.

Soit x^R une solution optimale du PR. Si x_i^R, $i \in J_B$, n'est pas entière, alors on peut exprimer sa partie fractionnaire comme suit :

$$x_i^R = \lfloor x_i^R \rfloor + f_i, \quad 0 < f_i < 1, \tag{2.41}$$

où f_i est la partie fractionnaire de x_i^R. La coupe de Gomory est alors donnée par :

$$-\sum_{j \in J_N} \left\{ \left(A_B^{-1}A\right)_{ij} \right\} x_j \leq -\left\{ \left(A_B^{-1}b\right)_i \right\}, \quad i \in J_B. \tag{2.42}$$

Algorithme 4 : Algorithme des coupes de Gomory

Entrées : *Le problème de PLNE*
Sorties : *Solution optimale* x^0*, valeur maximale* z^0

Relaxer le problème de PLNE pour obtenir le PR
Résoudre le PR pour obtenir la solution optimale x^R et la valeur maximale z^R de PR
tant que $x^R \notin \mathbb{Z}^n$ **faire**
    Sélectionner une variable x_i^R telle que $x_i^R \notin \mathbb{Z}$, $i \in J_B$
    Construire une coupe de Gomory et l'ajouter au PR
    Résoudre le nouveau problème pour obtenir x^R et z^R
fin tq
Poser $x^0 = x^R$ et $z^0 = z^R$
retourne x^0, z^0 ;

2.3.2.3 Heuristique d'arrondi

L'heuristique d'arrondi est une méthode approchée pour résoudre des problèmes de PLNE. Elle consiste à relaxer le problème (2.37), à résoudre le problème relaxé, puis à arrondir les composantes de la solution obtenue à des valeurs entières.

Il existe différentes manières d'effectuer cet arrondi, chacune ayant ses propres caractéristiques. Soit en effet x^R une solution optimale du problème relaxé. Pour tout $j \in J$, on a alors :

$$x_j^R = \lfloor x_j^R \rfloor + f_j, \quad 0 \leq f_j < 1. \tag{2.43}$$

2.3.2.3.1 Arrondi par rapport à la variable

Chaque composante x_j^R de la solution est arrondie à l'entier le plus proche. La solution arrondie est obtenue comme suit :

$$x_j^a = \begin{cases} \left\lfloor x_j^R \right\rfloor, & \text{si} \quad 0 \leq f_j < \dfrac{1}{2}, \\ \\ \left\lceil x_j^R \right\rceil, & \text{si} \quad \dfrac{1}{2} \leq f_j < 1. \end{cases} \tag{2.44}$$

Algorithme 5 : Arrondi par rapport à la variable

Entrées : *Le problème de PLNE*
Sorties : *Solution arrondie* x^a

`Relaxer PLNE pour obtenir le problème relaxé`
`Résoudre le PR pour obtenir la solution optimale` x^R
pour *tout* $j \in J$ **faire**
  si $0 \leq f_j < \frac{1}{2}$ **alors**
    $x_j^a = \left\lfloor x_j^R \right\rfloor$
  sinon
    $x_j^a = \left\lceil x_j^R \right\rceil$
  fin si
fin pour
retourne x^a ;

2.3.2.3.2 Arrondi par rapport à la fonction objectif

Toutes les composantes de la solution sont arrondies simultanément à la partie entière inférieure ou supérieure, en fonction de l'impact sur la fonction objectif. La solution arrondie est définie par :

$$x^a = \begin{cases} \left\lfloor x^R \right\rfloor & \text{si} \quad \sum_{j \in J} c_j f_j \geq 0, \\ \left\lceil x^R \right\rceil & \text{si} \quad \sum_{j \in J} c_j \leq \sum_{j \in J} c_j f_j < 0. \end{cases} \tag{2.45}$$

Algorithme 6 : Arrondi par rapport à la fonction objectif

Entrées : *Le problème de PLNE*
Sorties : *Solution arrondie* x^a

`Relaxer PLNE pour obtenir le PR`
`Résoudre le PR pour obtenir la solution optimale` x^R
si $\sum_{j \in J} c_j f_j \geq 0$ **alors**
  $x^a = \left\lfloor x^R \right\rfloor$
sinon
  si $\sum_{j \in J} c_j \leq \sum_{j \in J} c_j f_j < 0$ **alors**
    $x^a = \left\lceil x^R \right\rceil$
  fin si
fin si
retourne x^a ;

2.3.2.3.3 Arrondi par rapport au vecteur des coûts c

Chaque composante est arrondie à sa partie entière supérieure ou inférieure en fonction du signe du coefficient correspondant dans le vecteur des coûts c. La solution arrondie est donnée par :

$$x_j^a = \begin{cases} \left\lceil x_j^R \right\rceil, & \text{si} \quad c_j < 0, \\ \\ \left\lfloor x_j^R \right\rfloor, & \text{si} \quad c_j \geq 0. \end{cases} \tag{2.46}$$

Algorithme 7 : Arrondi par rapport au vecteur des coûts c

Entrées : *Le problème de PLNE*
Sorties : *Solution arrondie* x^a

```
Relaxer PLNE pour obtenir le problème relaxé (PR)
Résoudre le PR pour obtenir la solution optimale x^R
pour tout j ∈ J faire
    si c_j < 0 alors
        x_j^a = ⌈x_j^R⌉
    sinon
        x_j^a = ⌊x_j^R⌋
    fin si
fin pour
retourne x^a ;
```

Remarque 2.2. L'heuristique d'arrondi ne garantit pas de trouver une solution optimale, ni même une solution réalisable. En d'autres termes, l'arrondi peut parfois conduire à une solution non réalisable ou très éloignée de l'optimum. La qualité de la solution obtenue dépend du problème et de la solution optimale du problème relaxé.

Chapitre 3

Presolving

Sommaire

3.1 Introduction

Le presolving (en français prétraitement) est une phase préliminaire appliquée à un problème de Programmation Linéaire (PL) ou de Programmation Linéaire en Nombres Entiers (PLNE) avant de commencer la résolution. Il s'agit d'un ensemble d'opérations visant à simplifier le problème, en réduisant sa taille et sa complexité, tout en préservant l'optimalité de la solution. Ces simplifications consistent à fixer la valeur de certaines variables, à améliorer les bornes des variables et à supprimer des contraintes redondantes, ce qui permet de réduire l'ensemble de solutions réalisables. L'objectif principal du presolving est de simplifier le problème considéré et d'accélérer sa résolution, en réduisant le temps de calcul nécessaire pour trouver une solution optimale ou suboptimale.

Les techniques de réduction de variables ont été largement étudiées pour résoudre des problèmes d'optimisation NP-difficiles de grande taille. Les travaux [67, 68] ont proposé des méthodes de restriction des limites, d'élimination de lignes et de fixation de variables pour résoudre des problèmes de programmation linéaire en nombres entiers. L'article [5] a présenté une méthode basée sur la comparaison de paires de colonnes dans la matrice des contraintes, tandis que [73] a établi une condition nécessaire et suffisante pour identifier les variables réduites dans la matrice d'un problème général de programmation linéaire en nombres entiers. Cependant, la plupart de ces techniques traitent des problèmes d'optimisation linéaire avec des variables de décision non négatives, ce qui limite la possibilité de fixer uniquement une variable à zéro.

On trouve également des techniques de presolving pour les problèmes de programmation binaire, qui ont été étudiées par les auteurs [24, 41, 44, 46]. De plus, les techniques de presolving pour les problèmes de programmation mixte en nombres entiers ont été étudiées dans les travaux [3, 4, 23, 35, 34, 36, 51, 56, 58, 59]. Les techniques de presolving pour les programmes quadratiques convexes avec des variables continues et entières ont également été exposées dans [38, 45, 71], tandis que l'auteur [54] propose une technique de presolving pour les problèmes de programmation quadratique avec des contraintes simples de bornes. Enfin, les techniques de presolving pour les problèmes d'optimisation linéaire bi-niveaux ont été explorées dans [50].

La technique de réduction de variables proposée par [64] s'applique aux problèmes de programmation linéaire en nombres entiers avec des variables bornées. Elle permet de fixer certaines variables x_j du problème de PLNE à l'une de leurs bornes, 0 ou u_j, sans affecter l'optimalité. Les critères permettant d'identifier les variables pouvant être fixées sans affecter la solution optimale, sont définis en analysant les données du problème et en vérifiant certaines conditions.

Les techniques de presolving sont essentielles, car elles permettent aux solveurs de traiter des problèmes de grande taille de manière plus efficace. Les problèmes de PLNE, en particulier, peuvent devenir intraitables sans ces opérations.

Ce chapitre a pour but de présenter quelques techniques du presolving, d'explorer les différentes techniques utilisées et d'évaluer leur impact sur la résolution de problèmes PLNE.

3.2 Problème initial de programmation linéaire et le problème réduit

3.2.1 Problème initial

Nous considérons le problème de programmation linéaire en nombres entiers à variables bornées (PLNE) sous la forme suivante :

$$\begin{cases} \max & z(x) = c^T x \\ & Ax \leq b, \\ & l \leq x \leq u, \quad x \in \mathbb{Z}^n, \end{cases} \tag{3.1}$$

où

$$A = A(I,J) \in \mathbb{R}^{m \times n}, \quad I = \{1,2,\ldots,m\}, \quad J = \{1,2,\ldots,n\},$$

$$c = (c_1\ ,\ \ldots\ ,\ c_n)^T \in \mathbb{R}^n, \quad b = (b_1\ ,\ \ldots\ ,\ b_m)^T \in \mathbb{R}^m, \quad x = (x_1\ ,\ \ldots\ ,\ x_n)^T \in \mathbb{Z}^n,$$

$$l = (l_1\ ,\ \ldots\ ,\ l_n)^T \in \mathbb{Z}^n, \quad u = (u_1\ ,\ \ldots\ ,\ u_n)^T \in \mathbb{Z}^n, \quad l_j < u_j\ ,\ \forall j \in J.$$

Soit S le domaine réalisable du problème (3.1) :

$$S = \left\{ x = \left(x_j, j \in J\right) \in \mathbb{Z}^n\ :\ Ax \leq b\ ,\ l \leq x \leq u \right\}.$$

3.2.2 Problème réduit

Lorsqu'on applique le presolving à un problème de PLNE, on peut arriver à fixer certaines variables, supprimer des contraintes ou réduire les bornes, sans que ces modifications ne changent ni la solution optimale ni la valeur optimale. Le problème obtenu est alors dit problème réduit. Il est équivalent au problème initial, mais avec des dimensions réduites, ce qui le rend plus simple à résoudre.

Soit le problème réduit :

$$\begin{cases} \max & z_r(y) = \bar{c}^T y + \bar{z}, \\ & \bar{A}y \leq \bar{b}, \\ & \bar{l} \leq y \leq \bar{u}, \quad y \in \mathbb{Z}^{n_r}, \end{cases} \tag{3.2}$$

où $\bar{A} = A(I_r, J_r) \in \mathbb{R}^{m' \times n'}$, J_r est l'ensemble des indices des variables restantes après suppression des variables fixées ($|J_r| = n_r \leq n$) et I_r est l'ensemble des indices des contraintes restantes après suppression des contraintes redondantes ($|I_r| = m_r \leq m$), $y \in \mathbb{Z}^{n_r}$ représente les variables restantes après réduction, $\bar{c} = c(J_r) \in \mathbb{R}^{n'}$, $\bar{l} \geq l(J_r)$, $\bar{u} \leq u(J_r)$, $\bar{z} = c(J \setminus J_r)^T \tilde{x}(J \setminus J_r)$, $\tilde{x}(J \setminus J_r)$ représente les variables fixées, $\bar{b} = b(I_r) - A(I_r, J \setminus J_r)\tilde{x}(J \setminus J_r)$.

Le domaine réalisable du problème réduit est donné par :

$$S_r = \left\{ y \in \mathbb{Z}^{n'}\ :\ \bar{A}y \leq \bar{b},\ \bar{l} \leq y \leq \bar{u} \right\} \tag{3.3}$$

Définition 3.1. Soit x^0 la solution optimale du problème initial avec une valeur optimale $z(x^0)$, et y^0 la solution optimale du problème réduit avec une valeur optimale $z_r(y^0)$. Le problème réduit est équivalent au problème initial si et seulement si :

$$z(x^0) = z_r(y^0). \tag{3.4}$$

Dans ce cas, les variables fixées $\tilde{x}(J \setminus J_r)$ sont des composantes optimales. Le problème initial peut être remplacé par le problème réduit pour être résolu de manière plus efficace, et on a la relation :

$$x^0 = x^0(J) = \left(y^0(J_r)\ ,\ \tilde{x}(J \setminus J_r)\right). \tag{3.5}$$

3.3 Méthodes simples de presolving

Les méthodes simples de presolving consistent en une série de tests rapides permettant de détecter et de traiter des cas particuliers, tels que :

3.3.1 Ligne vide

Une ligne vide dans la matrice des contraintes correspond à une contrainte où tous les coefficients des variables sont nuls, c'est-à-dire que pour un certain $k \in I$, on a :

$$b_k \geq 0 \quad \text{et} \quad a_{kj} = 0, \quad \forall j \in J. \tag{3.6}$$

Dans ce cas, la contrainte est inutile, car elle n'a aucun impact sur l'espace des solutions admissibles. Elle peut donc être supprimée du problème.

3.3.2 Colonne vide

Une colonne vide correspond à une variable x_r qui n'apparaît dans aucune contrainte, c'est-à-dire pour un certain $r \in J$, on a :

$$a_{ir} = 0, \quad \forall i \in I. \tag{3.7}$$

Si x_r est également non liée à la fonction objectif ($c_r = 0$), elle est inutile et peut être retirée du problème. En revanche, si elle apparaît dans la fonction objectif, elle peut être fixée à sa borne supérieure $u_r < +\infty$ (si $c_r > 0$) ou sa borne inférieure $l_r > -\infty$ (si $c_r < 0$). Dans le cas contraire ($u_r = +\infty$ ou $l_r = -\infty$), le problème est non borné.

3.3.3 Variable irréalisable

Une variable irréalisable est une variable dont les bornes sont incohérentes ou contradictoires, c'est-à-dire que pour un certain $r \in J$, on a :

$$l_r > u_r. \tag{3.8}$$

Il devient alors impossible d'attribuer une valeur admissible à cette variable. Dans ce cas, le presolving peut identifier cette incohérence et indiquer immédiatement que le problème ne peut pas être résolu.

3.3.4 Variable fixée

Une variable fixée est une variable dont les bornes sont égales, c'est-à-dire :

$$l_r = u_r, \quad \text{pour un } r \in J. \tag{3.9}$$

Dans ce cas, la variable x_r peut être remplacée par sa valeur fixée dans le problème. Autrement dit, la variable optimale est $x_r^0 = l_r$.

3.4 Presolving dans les contraintes et les bornes

3.4.1 Contraintes irréalisables et redondances

Dans les problèmes de PLNE, les contraintes jouent un rôle essentiel pour définir l'espace des solutions admissibles. Cependant, il arrive souvent qu'un problème contienne des contraintes forcées, irréalisables et redondantes. Ces contraintes sont identifiées en exploitant les bornes des variables du problème.

3.4.1.1 Détection des contraintes irréalisables et redondances

Pour analyser les contraintes principales du problème (3.1), on calcule deux bornes $b_i^{\inf}$ et $b_i^{\sup}$ associées à chaque contrainte i :

$$b_i^{\inf} = \inf\{A_i^T x\} = \sum_{j \in J_i^+} a_{ij} l_j + \sum_{j \in J_i^-} a_{ij} u_j \quad \text{et} \quad b_i^{\sup} = \sup\{A_i^T x\} = \sum_{j \in J_i^+} a_{ij} u_j + \sum_{j \in J_i^-} a_{ij} l_j, \tag{3.10}$$

où $J_i^+ = \{j \in J \,:\, a_{ij} > 0\}$ et $J_i^- = \{j \in J \,:\, a_{ij} < 0\}$. Ainsi, pour tout $i \in I$ et pour tout x tel que $l \leq x \leq u$, on a :

$$b_i^{\inf} \leq \sum_{j \in J} a_{ij} x_j \leq b_i^{\sup}. \tag{3.11}$$

L'analyse de ces bornes révèle trois (03) cas possibles :

1. Les contraintes réalisables ;
2. Les contraintes irréalisables ;
3. Les contraintes redondantes.

3.4.1.2 Contraintes réalisables

Une contrainte pour un certain $i \in I$ est réalisable si :

$$b_i^{\inf} < b_i. \tag{3.12}$$

Cela signifie que la contrainte i est vérifiée pour les bornes des variables l et u.

3.4.1.3 Contraintes irréalisables

Une contrainte $i \in I$ est irréalisable si :

$$b_i < b_i^{\inf}. \tag{3.13}$$

Dans ce cas, il est impossible de satisfaire la contrainte i, ce qui rend le problème irréalisable.

3.4.1.4 Contraintes redondantes

Les contraintes redondantes sont des contraintes qui n'apportent aucune information supplémentaire. Ces contraintes augmentent inutilement la taille et la complexité du problème. Ainsi, elles ralentissent la résolution.

L'élimination des contraintes redondantes constitue une étape essentielle du presolving, visant à simplifier le problème sans affecter la solution optimale.

Une contrainte pour un certain $i \in I$ est redondante si :

$$b_i^{\sup} \leq b_i, \tag{3.14}$$

Dans ce cas, la $i^{\text{ème}}$ contrainte est toujours vérifiée.

3.4.2 Presolving dans les bornes

Le presolving dans les bornes consiste à resserrer les bornes inférieures l_j et supérieures u_j des variables avant de lancer la phase de résolution principale. L'objectif principal est de réduire l'espace de recherche en exploitant les relations entre les contraintes du problème et les bornes des variables. Il est aussi parfois possible de fixer certaines variables, simplifiant ainsi le problème et augmentant l'efficacité des algorithmes de résolution.

Cette technique est appliquée sur un problème exprimé sous forme standard. Nous considérons donc un problème formulé de la sorte :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax = b, \\ & l \leq x \leq u, \quad x \in \mathbb{Z}^n. \end{cases} \tag{3.15}$$

Lemme 3.1. *[4] Dans le problème* (3.15)*, supposons que la relation* $b_i^{\inf} \leq b_i \leq b_i^{\sup}$, $\forall i \in I$*, est vérifiée. Alors les nouvelles bornes d'une variable* x_r*,* $r \in J$ *sont calculées comme suit :*

$$\bar{l}_r = \max\left\{ l_r, \max_{i \in I} \{L_{ir}\} \right\}, \tag{3.16}$$

$$\bar{u}_r = \min\left\{ u_r, \min_{i \in I} \{U_{ir}\} \right\}, \tag{3.17}$$

où pour tout $i \in I$, les nombres L_{ir} et U_{ir} sont égaux à :

$$L_{ir} = \begin{cases} \left\lceil \dfrac{b_i - \sum\limits_{j \in J_i^+ \setminus \{r\}} a_{ij}u_j - \sum\limits_{j \in J_i^-} a_{ij}l_j}{a_{ir}} \right\rceil, & si \quad a_{ir} > 0, \\ \left\lceil \dfrac{b_i - \sum\limits_{j \in J_i^+} a_{ij}l_j - \sum\limits_{j \in J_i^- \setminus \{r\}} a_{ij}u_j}{a_{ir}} \right\rceil, & si \quad a_{ir} < 0. \end{cases} \tag{3.18}$$

$$U_{ir} = \begin{cases} \left\lfloor \dfrac{b_i - \sum\limits_{j \in J_i^+ \setminus \{r\}} a_{ij}l_j - \sum\limits_{j \in J_i^-} a_{ij}u_j}{a_{ir}} \right\rfloor, & si \quad a_{ir} > 0, \\ \left\lfloor \dfrac{b_i - \sum\limits_{j \in J_i^+} a_{ij}u_j - \sum\limits_{j \in J_i^- \setminus \{r\}} a_{ij}l_j}{a_{ir}} \right\rfloor, & si \quad a_{ir} < 0. \end{cases} \tag{3.19}$$

Démonstration. Supposons que $\bar{l}_r = L_{kr}$, pour un certain $k \in I$ et $r \in J$. Deux cas peuvent se présenter selon le signe de a_{kr} :

Cas 1. Si $a_{kr} > 0$, alors on a :

$$\bar{l}_r \leq x_r \Leftrightarrow L_{kr} \leq x_r \Leftrightarrow \left\lceil \frac{b_k - \sum\limits_{j \in J_k^+ \setminus \{r\}} a_{kj}u_j - \sum\limits_{j \in J_k^-} a_{kj}l_j}{a_{kr}} \right\rceil \leq x_r,$$

puisque $x_r \leq u_r$, on obtient :

$$\frac{b_k - \sum\limits_{j \in J_k^+ \setminus \{r\}} a_{kj}u_j - \sum\limits_{j \in J_k^-} a_{kj}l_j}{a_{kr}} \leq \left\lceil \frac{b_k - \sum\limits_{j \in J_k^+ \setminus \{r\}} a_{kj}u_j - \sum\limits_{j \in J_k^-} a_{kj}l_j}{a_{kr}} \right\rceil \leq x_r \leq u_r,$$

ainsi,

$$\begin{aligned} \frac{b_k - \sum\limits_{j \in J_k^+ \setminus \{r\}} a_{kj}u_j - \sum\limits_{j \in J_k^-} a_{kj}l_j}{a_{kr}} \leq u_r &\Leftrightarrow b_k - \sum_{j \in J_k^+ \setminus \{r\}} a_{kj}u_j - \sum_{j \in J_k^-} a_{kj}l_j \leq a_{kr}u_r \\ &\Leftrightarrow b_k \leq a_{kr}u_r + \sum_{j \in J_k^+ \setminus \{r\}} a_{kj}u_j + \sum_{j \in J_k^-} a_{kj}l_j \\ &\Leftrightarrow b_k \leq \sum_{j \in J_k^+} a_{kj}u_j + \sum_{j \in J_k^-} a_{kj}l_j, \end{aligned}$$

cela correspond à $b_k \leq b_k^{\sup}$.

Cas 2. Si $a_{kr} < 0$, nous obtenons :

$$\overline{l}_r \leq x_r \Leftrightarrow L_{kr} \leq x_r \Leftrightarrow \left\lceil \frac{b_k - \sum_{j \in J_k^+} a_{kj} l_j - \sum_{j \in J_k^- \setminus \{r\}} a_{kj} u_j}{a_{kr}} \right\rceil \leq x_r,$$

comme $x_r \leq u_r$, on déduit :

$$\frac{b_k - \sum_{j \in J_k^+} a_{kj} l_j - \sum_{j \in J_k^- \setminus \{r\}} a_{kj} u_j}{a_{kr}} \leq \left\lceil \frac{b_k - \sum_{j \in J_k^+} a_{kj} l_j - \sum_{j \in J_k^- \setminus \{r\}} a_{kj} u_j}{a_{kr}} \right\rceil \leq x_r \leq u_r,$$

cela implique :

$$\begin{aligned} \frac{b_k - \sum_{j \in J_k^+} a_{kj} l_j - \sum_{j \in J_k^- \setminus \{r\}} a_{kj} u_j}{a_{kr}} \leq u_r &\Leftrightarrow b_k - \sum_{j \in J_k^+} a_{kj} l_j - \sum_{j \in J_k^- \setminus \{r\}} a_{kj} u_j \geq a_{kr} u_r \\ &\Leftrightarrow b_k \geq a_{kr} u_r + \sum_{j \in J_k^+} a_{kj} l_j + \sum_{j \in J_k^- \setminus \{r\}} a_{kj} u_j \\ &\Leftrightarrow b_k \geq \sum_{j \in J_k^+} a_{kj} l_j + \sum_{j \in J_k^-} a_{kj} u_j, \end{aligned}$$

cela implique que $b_k \geq b_k^{\inf}$.

Cela montre que si $\overline{l}_r = L_{kr}$, alors les nouvelles bornes inférieures vérifie $b_k^{\inf} \leq b_k \leq b_k^{\sup}$.

De manière similaire, supposons que $\overline{u}_r = U_{kr}$. Nous obtenons selon le signe de a_{kr} :

Cas 1. Si $a_{kr} > 0$, alors on a :

$$x_r \leq \overline{u}_r \Leftrightarrow x_r \leq U_{kr} \Leftrightarrow x_r \leq \left\lfloor \frac{b_k - \sum_{j \in J_k^+ \setminus \{r\}} a_{kj} l_j - \sum_{j \in J_k^-} a_{kj} u_j}{a_{kr}} \right\rfloor,$$

comme $l_r \leq x_r$, on déduit :

$$l_r \leq x_r \leq \left\lfloor \frac{b_k - \sum_{j \in J_k^+ \setminus \{r\}} a_{kj} l_j - \sum_{j \in J_k^-} a_{kj} u_j}{a_{kr}} \right\rfloor \leq \frac{b_k - \sum_{j \in J_k^+ \setminus \{r\}} a_{kj} l_j - \sum_{j \in J_k^-} a_{kj} u_j}{a_{kr}},$$

donc, on aura :

$$\begin{aligned} l_r \leq \frac{b_k - \sum_{j \in J_k^+ \setminus \{r\}} a_{kj} l_j - \sum_{j \in J_k^-} a_{kj} u_j}{a_{kr}} &\Leftrightarrow a_{kr} l_r \leq b_k - \sum_{j \in J_k^+ \setminus \{r\}} a_{kj} l_j - \sum_{j \in J_k^-} a_{kj} u_j \\ &\Leftrightarrow a_{kr} l_r + \sum_{j \in J_k^+ \setminus \{r\}} a_{kj} l_j + \sum_{j \in J_k^-} a_{kj} u_j \leq b_k \\ &\Leftrightarrow \sum_{j \in J_k^+} a_{kj} l_j + \sum_{j \in J_k^-} a_{kj} u_j \leq b_k, \end{aligned}$$

cela correspond à $b_k^{\inf} \leq b_k$.

Cas 2. Si $a_{kr} < 0$, alors on a :

$$x_r \leq \overline{u}_r \Leftrightarrow x_r \leq U_{kr} \Leftrightarrow x_r \leq \left\lfloor \frac{b_k - \sum_{j \in J_k^+} a_{kj} u_j - \sum_{j \in J_k^- \setminus \{r\}} a_{kj} l_j}{a_{kr}} \right\rfloor,$$

comme $l_r \leq x_r$, on déduit :

$$l_r \leq x_r \leq \left\lfloor \frac{b_k - \sum_{j \in J_k^+} a_{kj} u_j - \sum_{j \in J_k^- \setminus \{r\}} a_{kj} l_j}{a_{kr}} \right\rfloor \leq \frac{b_k - \sum_{j \in J_k^+} a_{kj} u_j - \sum_{j \in J_k^- \setminus \{r\}} a_{kj} l_j}{a_{kr}},$$

donc, on aura :

$$\begin{aligned} l_r \leq \frac{b_k - \sum_{j \in J_k^+} a_{kj} u_j - \sum_{j \in J_k^- \setminus \{r\}} a_{kj} l_j}{a_{kr}} &\Leftrightarrow a_{kr} l_r \geq b_k - \sum_{j \in J_k^+} a_{kj} u_j - \sum_{j \in J_k^- \setminus \{r\}} a_{kj} l_j \\ &\Leftrightarrow a_{kr} l_r + \sum_{j \in J_k^+} a_{kj} u_j + \sum_{j \in J_k^- \setminus \{r\}} a_{kj} l_j \geq b_k \\ &\Leftrightarrow \sum_{j \in J_k^+} a_{kj} u_j + \sum_{j \in J_k^-} a_{kj} l_j \geq b_k, \end{aligned}$$

cela correspond à $b_k^{\sup} \geq b_k$.

Les nouvelles bornes supérieures $\overline{u}_r$ respectent les contraintes du problème et sont donc valides. □

Cependant, lorsque $\overline{l}_r = \overline{u}_r$, la variable x_r est fixée à

$$x_r^0 = \overline{l}_r = \overline{u}_r.$$

Dans ces cas, la variable x_r peut être retirée du problème initial.

3.4.3 Combinaison des techniques

Les techniques de presolving appliquées aux contraintes et aux bornes peuvent être combinées et itérées de manière alternée afin de maximiser leur impact. Cette approche consiste à appliquer successivement le resserrement des bornes des variables et l'analyse des contraintes pour identifier de nouvelles simplifications. Ce processus est répété jusqu'à ce que l'ensemble des solutions réalisables ne puisse plus être réduit.

Une telle combinaison itérative permet de détecter des interdépendances entre les contraintes et les bornes, souvent invisibles après une seule itération, et d'exploiter pleinement le potentiel de simplification du modèle.

Exemple 3.1. Considérons le problème suivant :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax \leq b, \\ & l \leq x \leq u \ , \ x \in \mathbb{N}^4. \end{cases}$$

où

$$A = \begin{pmatrix} 6 & -4 & 6 & 4 \\ -3 & -1 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 10 \\ 2 \\ 20 \end{pmatrix}, \quad c = \begin{pmatrix} 3 \\ 2 \\ -1 \\ 1 \end{pmatrix}, \quad l = \begin{pmatrix} 0 \\ 5 \\ 6 \\ 0 \end{pmatrix}, \quad u = \begin{pmatrix} 9 \\ 9 \\ 9 \\ 9 \end{pmatrix}.$$

Itération 1. — Nous calculons les bornes $b^{\inf}$ et $b^{\sup}$ des contraintes :

$$b^{\inf} = \begin{pmatrix} 6\times 0 - 4\times 9 + 6\times 6 + 4\times 0 \\ -3\times 9 - 1\times 9 + 2\times 6 + 2\times 0 \\ 1\times 0 + 1\times 5 + 1\times 6 + 1\times 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -24 \\ 11 \end{pmatrix},$$

$$b^{\sup} = \begin{pmatrix} 6\times 10 - 4\times 5 + 6\times 10 + 4\times 10 \\ -3\times 0 - 1\times 5 + 2\times 10 + 2\times 10 \\ 1\times 10 + 1\times 10 + 1\times 10 + 1\times 10 \end{pmatrix} = \begin{pmatrix} 124 \\ 31 \\ 36 \end{pmatrix}.$$

Les bornes calculées respectent la condition (3.12), ce qui signifie que les contraintes sont réalisables.

— Nous réécrivons le problème sous la forme standard :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax + I_3 y = b, \\ & l \leq x \leq u \ , \ x \in \mathbb{N}^4. \\ & 0 \leq y \leq u^e \ , \ y \in \mathbb{R}^3 \end{cases}$$

où les bornes $u^e = (10\ ,\ 26\ ,\ 9)^T$ sont calculées selon la formule (2.4).

— Nous calculons les nouvelles bornes, nous obtenons :

$$\begin{cases} L_{11} = \left\lceil \dfrac{10 + 4\times 5 - 6\times 9 - 4\times 9 - 1\times 10}{6} \right\rceil = -11, \\ L_{12} = \left\lceil \dfrac{2 + 1\times 9 - 2\times 5 - 2\times 0 - 1\times 0}{-3} \right\rceil = 1, \qquad \Rightarrow \bar{l}_1 := \max\{0, -11, 1, -16\} = 1, \\ L_{13} = \left\lceil \dfrac{20 - 1\times 9 - 1\times 9 - 1\times 9 - 1\times 9}{1} \right\rceil = -16, \end{cases}$$

$$\begin{cases} U_{11} = \left\lfloor \dfrac{10 + 4\times 9 - 6\times 6 - 4\times 0 - 1\times 0}{6} \right\rfloor = 1, \\ U_{12} = \left\lfloor \dfrac{2 + 1\times 5 - 2\times 9 - 2\times 9 - 1\times 26}{-3} \right\rfloor = 18, \quad \Rightarrow \bar{u}_1 := \min\{9, 1, 18, 9\} = 1, \\ U_{13} = \left\lfloor \dfrac{20 - 1\times 5 - 1\times 6 - 1\times 0 - 1\times 0}{1} \right\rfloor = 9, \end{cases}$$

$$\begin{cases} L_{21} = 7, \\ L_{22} = -17, \\ L_{23} = -16, \end{cases} \Rightarrow \bar{l}_2 := \max\{5,7,-17,-16\} = 7, \quad \begin{cases} U_{21} = 36, \\ U_{22} = 60, \\ U_{23} = 14, \end{cases} \Rightarrow \bar{u}_2 := \min\{9,36,60,14\} = 9,$$

$$\begin{cases} L_{31} = -11, \\ L_{32} = -18, \\ L_{33} = -16, \end{cases} \Rightarrow \bar{l}_3 := \max\{6,-11,-18,-16\} = 6, \quad \begin{cases} U_{31} = 7, \\ U_{32} = 19, \\ U_{33} = 15, \end{cases} \Rightarrow \bar{u}_3 := \min\{9,7,19,15\} = 7,$$

$$\begin{cases} L_{41} = -22, \\ L_{42} = -18, \\ L_{43} = -16, \end{cases} \Rightarrow \bar{l}_4 := \max\{0,-22,-18,-16\} = 0, \quad \begin{cases} U_{41} = 2, \\ U_{42} = 13, \\ U_{43} = 9, \end{cases} \Rightarrow \bar{u}_4 := \min\{9,2,13,9\} = 2.$$

Ainsi, comme $\bar{l}_1 = \bar{u}_1$, nous pouvons fixer $x_1^0 = 1$ et retirer cette variable du problème. Après cette réduction, nous obtenons pour $\bar{J} := \{2,3,4\}$:

$$\bar{A} := \begin{pmatrix} -4 & 6 & 4 \\ -1 & 2 & 2 \\ 1 & 1 & 1 \end{pmatrix}, \quad \bar{b} := b - a_1 x_1^0 = \begin{pmatrix} 4 \\ 5 \\ 19 \end{pmatrix}, \quad \bar{c} := \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}, \quad \bar{l} := \begin{pmatrix} 7 \\ 6 \\ 0 \end{pmatrix}, \quad \bar{u} := \begin{pmatrix} 9 \\ 7 \\ 2 \end{pmatrix} \quad \bar{z} = c_1 x_1^0 = 3.$$

Itération 2. — Calculons les bornes des contraintes :

$$b^{\inf} = \begin{pmatrix} 0 \\ 3 \\ 13 \end{pmatrix}, \quad b^{\sup} = \begin{pmatrix} 22 \\ 11 \\ 18 \end{pmatrix}.$$

Nous observons que $\overline{b}_3 < b_3$, ce qui indique que la 3$^{\text{ème}}$ contrainte est toujours satisfaite. Elle peut donc être retirée. Nous obtenons :

$$\bar{A} := \begin{pmatrix} -4 & 6 & 4 \\ -1 & 2 & 2 \end{pmatrix}, \quad \bar{b} := \begin{pmatrix} 4 \\ 5 \end{pmatrix}.$$

— Le nouveau problème sous forme standard devient :

$$\begin{cases} \max & z(X) = \bar{c}^T X + 3, \\ & \bar{A}X + I_2 y = \bar{b}, \\ & \bar{l} \leq X \leq \bar{u}\ ,\ X \in \mathbb{N}^3, \\ & 0 \leq y \leq u^e\ ,\ y \in \mathbb{R}^2. \end{cases}$$

où $X = x(J)$ et $u^e = (4\ ,\ 2)^T$.

— Recalculons les nouvelles bornes :

$$\begin{cases} L_{21} = 8, \\ L_{22} = 7, \end{cases} \Rightarrow \bar{l}_2 := \max\{7,8,7\} = 8, \quad \begin{cases} U_{31} = 6, \\ U_{32} = 7, \end{cases} \Rightarrow \bar{u}_3 := \min\{7,6,7\} = 6,$$

$$\begin{cases} U_{41} = 1, \\ U_{42} = 1, \end{cases} \Rightarrow \bar{u}_4 := \min\{2,1,1\} = 1.$$

Ainsi, $x_3^0 = 6$, car $\bar{l}_3 = \bar{u}_3$. Nous obtenons :

$$\bar{A} := \begin{pmatrix} -4 & 4 \\ -1 & 2 \end{pmatrix}, \quad \bar{b} := \bar{b} - a_3 x_3^0 = \begin{pmatrix} -32 \\ -7 \end{pmatrix}, \quad \bar{c} := \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \quad \bar{l} := \begin{pmatrix} 8 \\ 0 \end{pmatrix}, \quad \bar{u} := \begin{pmatrix} 9 \\ 1 \end{pmatrix}, \quad \bar{z} := \bar{z} + c_3 x_3^0 = -3.$$

Nous trouvons que la solution optimale du problème réduit est $x_2^0 = 9$ et $x_4^0 = 1$. Alors la solution optimale du problème initial est $x^0 = (1\ ,\ 9\ ,\ 6\ ,\ 1)^T$, avec $z^0 = z(x^0) = 16$.

3.5 Presolving dans les problèmes de PL à variables non négatives

Dans cette section, nous allons considérer le problème de PLNE où $l_j = 0$ et $u_j = +\infty$, $\forall j \in J$. Le problème est donc défini comme suit :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax \leq b, \\ & x \in \mathbb{N}^n. \end{cases} \tag{3.20}$$

Soit S_∞ le domaine réalisable du problème (3.20)

$$S_\infty = \{x \in \mathbb{N}^n \ : \ Ax \leq b\}. \tag{3.21}$$

Définition 3.2 (Variable redondante)**.** Dans un problème de PLNE à variables non négatives, une colonne r est supprimée si la valeur de la solution optimale (si elle existe) ne change pas lorsque r est retiré de l'ensemble des indices J. Autrement dit, la variable correspondante x_r peut être fixée à zéro sans affecter la solution optimale du problème. Cette variable x_r est appelée variable redondante (ou variable dominée).

Dans cette section, nous étudierons les méthodes permettant de détecter ces variables et de les supprimer du problème pour améliorer l'efficacité de la résolution.

Théorème 3.1. *[72] Dans le problème de PLNE à variables non négatives, la variable x_r, $r \in J$, est retirée s'il existe des entiers non négatifs p_j tels que*

$$\sum_{j \in J_r} c_j p_j \geq c_r, \quad et \quad \sum_{j \in J_r} a_{ij} p_j \leq a_{ir}, \quad \forall i \in I. \tag{3.22}$$

où $J_r = J \setminus \{r\}$.

Démonstration. Soit $\tilde{x}$ une SR du problème (3.20), i.e. $\tilde{x} \in S_\infty$. Soit $\bar{x} \geq 0$ tel que :

$$\begin{cases} \bar{x}_j = \tilde{x}_j + p_j \tilde{x}_r, \quad \forall j \in J_r, \\ \bar{x}_r = 0. \end{cases}$$

On a alors :

$$\begin{aligned} \sum_{j \in J} a_{ij} \bar{x}_j &= \sum_{j \in J_r} a_{ij} \bar{x}_j + \cancel{a_{ir} \bar{x}_r} = \sum_{j \in J_r} a_{ij} (\tilde{x}_j + p_j \tilde{x}_r) = \sum_{j \in J_r} a_{ij} \tilde{x}_j + \tilde{x}_r \sum_{j \in J_r} a_{ij} p_j \\ &\leq \sum_{j \in J_r} a_{ij} \tilde{x}_j + a_{ir} \tilde{x}_r = \sum_{j \in J} a_{ij} \tilde{x}_j \\ &\leq b_i, \quad \forall i \in I, \end{aligned}$$

d'où $\bar{x}$ satisfait les contraintes du problème (3.20). De plus, on déduit :

$$z(\bar{x}) = \sum_{j\in J} c_j\bar{x}_j = \sum_{j\in J_r} c_j\bar{x}_j + \cancel{c_r\bar{x}_r} = \sum_{j\in J_r} c_j(\tilde{x}_j + p_j\tilde{x}_r) = \sum_{j\in J_r} c_j\tilde{x}_j + \tilde{x}_r \sum_{j\in J_r} c_j p_j$$
$$\geq \sum_{j\in J_r} c_j\tilde{x}_j + c_r\tilde{x}_r = \sum_{j\in J} c_j\tilde{x}_j = z(\tilde{x}),$$

c'est-à-dire que la nouvelle solution $\bar{x}$ est meilleure ou aussi bonne que $\tilde{x}$. Ainsi, si $\tilde{x}$ est optimale, il existe toujours une autre solution optimale $\bar{x}$ telle que $\bar{x}_r = 0$. Par conséquent, la variable x_r est redondante. □

Lemme 3.2. *[72] Dans le problème* (3.20)*, étant donnée une paire arbitraire* $(q,r),\ 1 \leq q \neq r \leq n$*, s'il existe un entier non négatif* p_q *tel que*

$$c_q p_q \geq c_r \quad et \quad a_{iq}p_q \leq a_{ir}, \quad \forall i \in I, \tag{3.23}$$

alors la variable x_r *est fixée à* 0*.*

Démonstration. En utilisant le théorème (3.1), et en prenant $p_j = 0,\ j \in J_r\backslash\{q\}$. On obtient :

$$\sum_{j\in J_r} a_{ij}p_j = a_{iq}p_q \leq a_{ir},$$
$$\sum_{j\in J_r} c_j p_j = c_q p_q \geq c_r.$$

D'où $x_r^0 = 0$. □

Afin de démontrer le théorème suivant, nous introduisons les notations suivantes. Pour un couple (q,r) tel que $1 \leq q \neq r \leq n$, on note :

$$\begin{cases} I_1 = I_1(q,r) = \{i \in I : a_{ir} \geq 0 \text{ et } a_{iq} > 0\}, \\ I_2 = I_2(q,r) = \{i \in I : a_{ir} \leq 0 \text{ et } a_{iq} < 0\}, \\ I_3 = I_3(q,r) = \{i \in I : a_{ir} < 0 \text{ et } a_{iq} \geq 0\}, \\ I_4 = I_4(q,r) = I\backslash(I_1 \cup I_2 \cup I_3), \end{cases}$$

où $I_1 \cup I_2 \cup I_3 \cup I_4 = I$ et $I_{k_1} \cap I_{k_2} = \emptyset$, $1 \leq k_1, k_2 \leq 4$, $k_1 \neq k_2$. Soit :

$$p_q = \begin{cases} \min\limits_{i\in I_1} \left\lfloor \dfrac{a_{ir}}{a_{iq}} \right\rfloor, & \text{si} \quad I_1 \neq \emptyset, \\ \\ \max\limits_{i\in I_2} \left\lceil \dfrac{a_{ir}}{a_{iq}} \right\rceil, & \text{si} \quad I_2 \neq \emptyset, \end{cases} \tag{3.24}$$

Théorème 3.2. *[72] Supposons que pour une paire arbitraire* $(q,r), 1 \leq q \neq r \leq n$*, l'une de ces conditions suivantes est remplie :*

$$(i) \qquad I_1 \neq \emptyset,\ I_2 \cup I_3 = \emptyset, \qquad et \quad c_q p_q \geq c_r, \tag{3.25}$$

$$(ii) \qquad I_2 \neq \emptyset,\ I_1 \cup I_3 = \emptyset, \qquad et \quad c_q p_q \geq c_r, \tag{3.26}$$

$$(iii) \qquad I_1 \neq \emptyset,\ I_2 \neq \emptyset,\ I_3 = \emptyset, \qquad \min_{i \in I_1} \left\lfloor \frac{a_{ir}}{a_{iq}} \right\rfloor \geq \max_{i \in I_2} \left\lceil \frac{a_{ir}}{a_{iq}} \right\rceil \quad et \quad c_q p_q \geq c_r, \tag{3.27}$$

$$(iv) \qquad I_1 \cup I_2 \cup I_3 = \emptyset, \qquad c_q > 0 \quad et \quad c_r > 0, \tag{3.28}$$

$$(v) \qquad I_1 \cup I_2 \cup I_3 = \emptyset, \qquad et \quad c_r \leq 0. \tag{3.29}$$

Alors x_r est une variable dominée.

Démonstration. $(i) *$ Si $I_1 \neq \emptyset$, $I_2 = \emptyset$ et $I_3 = \emptyset$, d'après (3.24) on a :

$$p_q = \min_{i \in I_1} \left\lfloor \frac{a_{ir}}{a_{iq}} \right\rfloor \geq 0.$$

$$\begin{aligned} \forall i \in I_1 : p_q = \min_{i \in I_1} \left\lfloor \frac{a_{ir}}{a_{iq}} \right\rfloor &\Rightarrow p_q \leq \left\lfloor \frac{a_{ir}}{a_{iq}} \right\rfloor \\ &\Rightarrow p_q \leq \frac{a_{ir}}{a_{iq}} \\ &\Rightarrow a_{iq} p_q \leq a_{ir}, \end{aligned} \tag{3.30}$$

$$\forall i \in I_4 : \begin{cases} a_{iq} \leq 0 \quad \text{et} \quad a_{ir} > 0 \\ \qquad\quad \text{ou} \\ a_{iq} = 0 \quad \text{et} \quad a_{ir} = 0 \end{cases} \Leftrightarrow \begin{cases} a_{iq} p_q < a_{ir} \\ \quad \text{ou} \\ a_{iq} p_q = a_{ir} \end{cases} \tag{3.31}$$

$$\Leftrightarrow a_{iq} p_q \leq a_{ir},$$

d'après (3.30) et (3.31), ainsi que $I_2 = I_3 = \emptyset$, on aura

$$a_{iq} p_q \leq a_{ir} \quad \forall i \in I.$$

Avec la condition $c_q p_q \geq c_r$ de la relation (3.25), on conclut d'après le lemme 3.2 que la variable x_r est dominée.

$(ii) *$ Si $I_1 = \emptyset$, $I_2 \neq \emptyset$ et $I_3 = \emptyset$, d'après (3.24) on a :

$$p_q = \max_{i \in I_2} \left\lceil \frac{a_{ir}}{a_{iq}} \right\rceil \geq 0.$$

$$\begin{aligned} p_q = \max_{i \in I_2} \left\lceil \frac{a_{ir}}{a_{iq}} \right\rceil &\Rightarrow p_q \geq \left\lceil \frac{a_{ir}}{a_{iq}} \right\rceil \\ &\Rightarrow p_q \geq \frac{a_{ir}}{a_{iq}} \\ &\Rightarrow a_{iq} p_q \leq a_{ir}, \quad \forall i \in I_2. \end{aligned} \tag{3.32}$$

D'autre part, d'après (3.32), on a pour tout $i \in I_4$:

$$a_{iq} p_q \leq a_{ir}. \tag{3.33}$$

D'après (3.32) et (3.33), ainsi que $I_1 = I_3 = \emptyset$, on obtient

$$a_{iq} p_q \leq a_{ir} \quad \forall i \in I.$$

Avec la condition $c_q p_q \geq c_r$ de la relation (3.26), on déduit que la variable x_r est donc dominée.

(*iii*) $*$ Si $I_1 \neq \emptyset$, $I_2 \neq \emptyset$ et $I_3 = \emptyset$, alors on pose :

$$p_q^1 = \min_{i \in I_1} \left\lfloor \frac{a_{ir}}{a_{iq}} \right\rfloor \geq 0 \text{ et } p_q^2 = \max_{i \in I_2} \left\lceil \frac{a_{ir}}{a_{iq}} \right\rceil \geq 0,$$

et d'après (*iii*), on a

$$p_q^1 \geq p_q^2.$$

En posant $p_q = p_q^1$, on déduit :

$$\forall i \in I_1 : p_q = p_q^1 \Leftrightarrow a_{iq} p_q \leq a_{ir}, \tag{3.34}$$

$$\begin{aligned} \forall i \in I_2 : p_q \geq p_q^2 &\Leftrightarrow a_{iq} p_q \leq a_{iq} p_q^2 \leq a_{ir} \\ &\Leftrightarrow a_{iq} p_q \leq a_{ir}, \end{aligned} \tag{3.35}$$

$$\forall i \in I_4 : a_{iq} p_q \leq a_{ir}, \tag{3.36}$$

d'où :

$$a_{iq} p_q \leq a_{ir},\ i \in I.$$

D'après le lemme 3.2, la variable x_r est donc dominée.

(*iv*) $*$ Si $I_1 \cup I_2 \cup I_3 = \emptyset$, alors pour $i \in I_4$ la condition $a_{iq} p_q \leq a_{ir}$ est toujours satisfaite pour tout nombre entier $p_q \geq 0$. On peut poser $p_q = \left\lceil \frac{c_r}{c_q} \right\rceil$ pour réaliser $c_q p_q \geq c_r$. D'après le lemme 3.2, la variable x_r est donc dominée.

(*v*) $*$ Si $I_1 \cup I_2 \cup I_3 = \emptyset$, pour $i \in I_4$ la condition $a_{iq} p_q \leq a_{ir}$ est toujours satisfaite pour tout nombre entier $p_q \geq 0$. Si $c_q \geq 0$, alors $c_q p_q \geq c_r$, car $c_r \leq 0$ par hypothèse. Dans le cas où $c_q < 0$, on peut poser $p_q = \left\lfloor \frac{c_r}{c_q} \right\rfloor$ pour réaliser $c_q p_q \geq c_r$. D'après le lemme 3.2, la variable x_r est donc dominée.

□

Exemple 3.2. On considère le problème suivant :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax \leq b, \\ & x \geq 0, \quad x \in \mathbb{N}^6, \end{cases}$$

où

$$A = \begin{pmatrix} 13 & 29 & 5 & 7 & 4 & 2 \\ -9 & 4 & -5 & 5 & 9 & 5 \\ 7 & 5 & -3 & 9 & -8 & 6 \end{pmatrix}, \quad b = \begin{pmatrix} 28 \\ 68 \\ 66 \end{pmatrix}, \quad c = \begin{pmatrix} -5 \\ 19 \\ 19 \\ 5 \\ 14 \\ 20 \end{pmatrix}, \quad I = \{1,2,3\}, \quad J = \{1,2,3,4,5,6\}.$$

- Pour $q = 3$, $r = 1$, on obtient :

$$I_1(3,1) = \{1\},\ I_2(3,1) = \{2\},\ I_3(3,1) = \emptyset,\ I_4(3,1) = \{3\},$$
$$p_3^1 = \left\lfloor \frac{13}{5} \right\rfloor = 2 \geq p_3^2 = \left\lceil \frac{-9}{-5} \right\rceil = 2 \Rightarrow p_3 = 2,$$
$$c_3 p_3 = 38 > c_1 = -5,$$

  d'après le théorème 3.2, $x_1^0 = 0$.

- Pour $q = 6$, $r = 4$, on a

$$I_1(6,4) = \{1,2,3\},\ I_2(6,4) = I_3(6,4) = I_4(6,4) = \emptyset,$$
$$p_6 = \min\left\{ \left\lfloor \frac{7}{2} \right\rfloor, \left\lfloor \frac{5}{5} \right\rfloor, \left\lfloor \frac{9}{6} \right\rfloor \right\} = 1,$$
$$c_6 p_6 = 20 > c_4 = 5,$$

  d'après le théorème (3.2), x_4 est une variable dominée.

On pose $y_1 = x_2$, $y_2 = x_3$, $y_3 = x_5$ et $y_4 = x_6$. Alors le nouveau problème s'écrit comme suit :

$$\begin{cases} \max & z_r(y) = \bar{c}^T y, \\ & \bar{A} y \leq \bar{b}, \\ & y \geq 0, \quad y \in \mathbb{N}^4, \end{cases}$$

où

$$\bar{A} = \begin{pmatrix} 29 & 5 & 4 & 2 \\ 4 & -5 & 9 & 5 \\ 5 & -3 & -5 & 6 \end{pmatrix}, \quad \bar{b} = \begin{pmatrix} 28 \\ 68 \\ 66 \end{pmatrix}, \quad \bar{c} = \begin{pmatrix} 19 \\ 19 \\ 14 \\ 10 \end{pmatrix}.$$

La solution optimale du problème réduit est $y^0 = (0\ ,\ 1\ ,\ 0\ ,\ 11)^T$, avec $z_r(y^0) = 129$.

Donc, pour le problème initial, la solution optimale est $x^0 = (0\ ,\ 0\ ,\ 1\ ,\ 0\ ,\ 0\ ,\ 11)^T$, avec $z(x^0) = 129$.

Remarque 3.1. les deux problèmes ci-dessus ont été implémentés sous MATLAB R2021b, et ils ont conduit à la même solution optimale ainsi qu'à la même valeur maximale. Cela indique que le presolving a fixé les variables à leurs valeurs optimales.

3.6 Presolving dans les problèmes de PLNB

Dans cette partie, nous appliquons certaines techniques de presolving au problème de PLNB. Nous considérons dans le problème de PLNE que les variables sont bornées entre $l_j = 0$ et $u_j = 1$, $\forall j \in J$. Cela nous permet d'obtenir un problème de PLNB sous la forme suivante :

$$\begin{cases} \max & z(x) = c^T x \\ & Ax \leq b, \\ & x \in \{0,1\}^n. \end{cases} \tag{3.37}$$

Le domaine réalisable du problème de PLNB est donnée par :

$$S_1 = \{x \in \{0,1\}^n \ : \ Ax \leq b\}. \tag{3.38}$$

Lemme 3.3. *Dans le PLNB, si l'on trouve $k \in I$ et $r \in J$ tels que :*

$$a_{kr} > b_k, \quad et \quad a_{kj} > 0, \quad \forall j \in J_r, \tag{3.39}$$

alors $x_r^0 = 0$ est une composante optimale.

Démonstration. Supposons que x^0 soit une solution optimale, avec $x_r^0 = 1$. Dans ce cas, pour $i = k$, nous aurons :

$$\sum_{j \in J} a_{kj} x_j^0 \leq b_k \quad \Rightarrow \quad a_{kr} x_r^0 \leq b_k \quad \Rightarrow \quad a_{kr} \leq b_k,$$

puisque $x_r^0 = 1$. Cette dernière inégalité contredit l'hypothèse $a_{kr} > b_k$. Par conséquent, aucune solution optimale ne peut satisfaire $x_r^0 = 1$, ce qui implique que $x_r^0 = 0$. □

Lemme 3.4. *Dans PLNB, si $r \in J$ vérifie :*

$$a_r \geq 0 \quad et \quad c_r \leq 0, \tag{3.40}$$

alors $x_r^0 = 0$.

Démonstration. Supposons que le vecteur $\bar{x} \in S_1$ est une solution réalisable de (PLNB) avec $\bar{x}_r = 1$. Définissons un vecteur $\hat{x}$ tel que :

$$\begin{cases} \hat{x}_j = \bar{x}_j, \quad \forall j \in J_r, \\ \hat{x}_r = 0. \end{cases}$$

Pour tout $i \in I$, nous avons :

$$\begin{aligned} \sum_{j \in J} a_{ij} \hat{x}_j &= \sum_{j \in J_r} a_{ij} \hat{x}_j + \cancel{a_{ir} \hat{x}_r} \\ &\leq \sum_{j \in J_r} a_{ij} \bar{x}_j + a_{ir} \bar{x}_r = \sum_{j \in J} a_{ij} \bar{x}_j \\ &\leq b_i, \end{aligned}$$

donc $\hat{x}$ satisfait les contraintes du problème de (PLNB), ce qui implique $\hat{x} \in S_1$. En outre, pour la fonction objectif, nous avons :

$$\begin{aligned} z(\hat{x}) = \sum_{j \in J} c_j \hat{x}_j &= \sum_{j \in J_r} c_j \hat{x}_j + \cancel{c_r \hat{x}_r} \\ &\geq \sum_{j \in J_r} c_j \bar{x}_j + c_r \bar{x}_r = \sum_{j \in J} c_j \bar{x}_j = z(\bar{x}), \end{aligned}$$

c'est-à-dire que la nouvelle solution $\hat{x}$ est meilleure ou aussi bonne que $\bar{x}$. Par conséquent, il est toujours possible de trouver une solution optimale x^0, avec $x_r^0 = 0$. □

Remarque 3.2. Il est intéressant de noter que Zhu et al. [72] ont également posé une hypothèse similaire à celle que nous avons formulée dans la relation (3.40). Cependant, la différence majeure entre ces deux propositions réside dans le fait que Zhu et al. l'ont appliquée à un problème qui n'était pas borné, tandis que la nôtre est appliquée à un problème borné. De plus, une autre distinction importante entre les deux hypothèses réside dans la manière dont nous traitons la colonne r, où dans l'hypothèse de Zhu et al., la colonne r est dominée par une autre colonne du problème, tandis que dans notre cas, nous nous intéressons uniquement à la colonne r.

Théorème 3.3. *Dans le problème de programmation linéaire en nombres binaires (PLNB), la variable* x_r, $r \in J$, *est fixée à* 1 *si :*

$$a_r \leq 0 \quad et \quad c_r \geq 0. \tag{3.41}$$

Démonstration. Supposons que le vecteur $\bar{x} \in S_1$ soit une solution réalisable de (PLNB), avec $\bar{x}_r = 0$, et définissons un vecteur $\hat{x}$ tel que :

$$\begin{cases} \hat{x}_j = \bar{x}_j, & j \in J_r, \\ \hat{x}_r = 1. \end{cases}$$

Nous avons alors :

$$\begin{aligned} \sum_{j \in J} a_{ij}\hat{x}_j &= \sum_{j \in J_r} a_{ij}\hat{x}_j + a_{ir}\hat{x}_r \\ &\leq \sum_{j \in J_r} a_{ij}\bar{x}_j + \cancel{a_{ir}\bar{x}_r} = \sum_{j \in J} a_{ij}\bar{x}_j \\ &\leq b_i, \quad \forall i \in I. \end{aligned}$$

Ainsi, $\hat{x}$ satisfait les contraintes du problème de PLNB, i.e, $\hat{x} \in S_1$. De plus, pour la fonction objectif, nous déduisons que :

$$\begin{aligned} z(\hat{x}) &= \sum_{j \in J} c_j\hat{x}_j = \sum_{j \in J_r} c_j\hat{x}_j + c_r\hat{x}_r \\ &\geq \sum_{j \in J_r} c_j\bar{x}_j + \cancel{c_r\bar{x}_r} = \sum_{j \in J} c_j\bar{x}_j = z(\bar{x}). \end{aligned}$$

Cela signifie que la nouvelle solution $\hat{x}$ est meilleure ou aussi bonne que $\bar{x}$. Ainsi, si $\bar{x}$ est optimale, il existe toujours une autre solution optimale x^0 telle que $x^0_r = 1$. □

Théorème 3.4. *Dans le problème de programmation linéaire en nombres binaires (PLNB), la variable* x_r, $r \in J$, *est fixée à* 1 *si :*

$$\begin{cases} \max\limits_{j \in J}\{c_j\} = c_r \geq 0, \\ \qquad\qquad et \\ \min\limits_{j \in J}\{a_{ij}\} = a_{ir}, \quad avec \quad a_{ir} \leq b_i, \quad \forall i \in I. \end{cases} \tag{3.42}$$

Démonstration. Nous avons deux cas à considérer, qui sont les suivants :

i) Supposons que le vecteur $\bar{x}$ est une solution réalisable de (PLNB) avec $\bar{x}_r = 0$ et $\bar{x}_q = 1$, $q \in J_r$, est un indice arbitraire. Soit $\hat{x}$ tel que :

$$\begin{cases} \hat{x}_j = \bar{x}_j, \quad \forall j \in J \backslash \{r,q\}, \\ \hat{x}_r = 1, \\ \hat{x}_q = 0. \end{cases}$$

D'après (3.42), nous avons :

$$\begin{aligned} \sum_{j \in J} a_{ij}\hat{x}_j &= \sum_{j \in J \backslash \{r,q\}} a_{ij}\hat{x}_j + a_{ir}\hat{x}_r + \cancel{a_{iq}\hat{x}_q} \\ &\leq \sum_{j \in J \backslash \{r,q\}} a_{ij}\bar{x}_j + a_{iq}\bar{x}_q + \cancel{a_{ir}\bar{x}_r} = \sum_{j \in J} a_{ij}\bar{x}_j \\ &\leq b_i, \quad \forall i \in I, \end{aligned}$$

donc $\hat{x}$ vérifie les contraintes de (PLNB), i.e, $\hat{x} \in S_1$. En outre, nous déduisons que :

$$\begin{aligned} z(\hat{x}) = \sum_{j \in J} c_j\hat{x}_j &= \sum_{j \in J \backslash \{r,q\}} c_j\hat{x}_j + c_r\hat{x}_r + \cancel{c_q\hat{x}_q} \\ &\geq \sum_{j \in J \backslash \{r,q\}} c_j\bar{x}_j + c_q\bar{x}_q + \cancel{c_r\bar{x}_r} = \sum_{j \in J} c_j\bar{x}_j = z(\bar{x}), \end{aligned}$$

c'est-à-dire que la nouvelle solution $\hat{x}$ est meilleure ou aussi bonne que $\bar{x}$.

ii) Supposons que le vecteur $\bar{x} \in S_1$ est une solution réalisable de (PLNB) avec $\bar{x}_r = 0$, et il n'existe pas d'indice $q \in J_r$ tel que $\bar{x}_q = 1$, donc $\bar{x} = 0$. Soit alors $\hat{x}$ tel que :

$$\begin{cases} \hat{x}_j = \bar{x}_j, \quad \forall j \in J_r, \\ \hat{x}_r = 1. \end{cases}$$

D'après (3.42), nous avons :

$$\sum_{j \in J} a_{ij}\hat{x}_j = a_{ir}\hat{x}_r = a_{ir} \leq b_i, \quad \forall i \in I,$$

donc $\hat{x}$ vérifie les contraintes de (PLNB), i.e, $\hat{x} \in S_1$. En outre, nous déduisons que :

$$z(\hat{x}) = \sum_{j \in J} c_j\hat{x}_j = c_r\hat{x}_r = c_r \geq 0 = \sum_{j \in J} c_j\bar{x}_j = z(\bar{x}),$$

c'est-à-dire que la nouvelle solution $\hat{x}$ est meilleure ou aussi bonne que $\bar{x}$.

Ainsi, d'après les cas possibles i) et ii), nous pouvons toujours trouver une solution optimale x^0, avec $x^0_r = 1$. □

3.7 Presolving dans les problèmes de PLNE à variables bornées

Dans cette section, nous allons considérer le problème de PLNE où $l_j = 0$ et $u_j > 0$, $\forall j \in J$. Le problème s'écrit donc comme suit :

$$\begin{cases} \max & z(x) = c^T x \\ & Ax \leq b, \\ & x \in \mathbb{N}^n, \end{cases} \tag{3.43}$$

Soit S^+ le domaine réalisable du problème (3.43) :

$$S^+ = \{x \in \mathbb{N}^n \ : \ Ax \leq b, \ 0 \leq x \leq u\}. \tag{3.44}$$

3.7.1 Reformulation du problème de PLNE en un problème de PLNB

Nous nous intéressons principalement dans notre étude théorique à la reformulation du problème (3.43) sous forme d'un programme linéaire en nombres binaires, où nous remplaçons une variable entière x_j par une combinaison linéaire de variables binaires $t_{j1}, t_{j2}, \ldots, t_{ju_j}$ telle que :

$$x_j = \sum_{k=1}^{u_j} (t_{jk} \times k), \text{ avec } t_{jk} \in \{0,1\}. \tag{3.45}$$

Pour garantir que la variable x_j est bien représentée par la combinaison linéaire de variables binaires, les contraintes du problème initial sont modifiées en ajoutant des contraintes supplémentaires telles que :

$$\sum_{k=1}^{u_j} t_{jk} \leq 1. \tag{3.46}$$

La nouvelle formulation du problème s'écrit alors :

$$\begin{cases} \max & \xi(t) = d^T t \\ & Bt \leq b, \\ & \sum_{k=1}^{u_j} t_{jk} \leq 1, \quad \forall j \in J, \\ & t = (t_{11} \ , \ \ldots \ , \ t_{1u_1} \ ; \ \ldots \ ; \ t_{n1} \ , \ \ldots \ , \ t_{nu_n})^T, \quad t_{jk} \in \{0,1\}, \end{cases} \tag{3.47}$$

où :

$$d = (c_1 \ , \ 2c_1 \ , \ \ldots \ , \ u_1c_1 \ ; \ c_2 \ , \ 2c_2 \ , \ \ldots \ , \ u_2c_2 \ ; \ \ldots \ ; \ c_n \ , \ \ldots \ , \ u_nc_n)^T \in \mathbb{R}^{\bar{n}},$$

$$B = (a_1 \ , \ 2a_1 \ , \ \ldots \ , \ u_1a_1 \ ; \ a_2 \ , \ 2a_2 \ , \ \ldots \ , \ u_2a_2 \ ; \ \ldots \ ; \ a_n \ , \ \ldots \ , \ u_na_n) \in \mathbb{R}^{m \times \bar{n}},$$

avec : $\bar{n} = \sum_{j \in J} u_j$.

Exemple 3.3. Nous considérons le problème suivant :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax \leq b, \\ & 0 \leq x \leq u, \quad x \in \mathbb{N}^3, \end{cases}$$

où

$$A = \begin{pmatrix} 2 & 1 & 4 \\ 3 & -2 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 12 \\ 10 \end{pmatrix}, \quad c = \begin{pmatrix} -2 \\ 3 \\ 2 \end{pmatrix}, \quad u = \begin{pmatrix} 2 \\ 3 \\ 3 \end{pmatrix}.$$

Nous pouvons exprimer les variables x_1, x_2 et x_3 en termes de variables binaires t_{jk}, comme suit :

$$\begin{cases} x_1 = \sum_{k=1}^{2} (t_{1k} \times k) = t_{11} + 2t_{12}, & t_{11}, t_{12} \in \{0,1\}, \\ x_2 = \sum_{k=1}^{3} (t_{2k} \times k) = t_{21} + 2t_{22} + 3t_{23}, & t_{21}, t_{22}, t_{23} \in \{0,1\}, \\ x_3 = \sum_{k=1}^{3} (t_{3k} \times k) = t_{31} + 2t_{32} + 3t_{33}, & t_{31}, t_{32}, t_{33} \in \{0,1\}, \end{cases}$$

avec :

$$\begin{cases} t_{11} + t_{12} \leq 1, \\ t_{21} + t_{22} + t_{23} \leq 1, \\ t_{31} + t_{32} + t_{33} \leq 1. \end{cases}$$

En substituant ces expressions dans le problème, nous obtenons :

$$\begin{aligned} z(x) = -2x_1 + 3x_2 + 2x_3 &= -2(t_{11} + 2t_{12}) + 3(t_{21} + 2t_{22} + 3t_{23}) + 2(t_{31} + 2t_{32} + 3t_{33}) \\ &= -2t_{11} - 4t_{12} + 3t_{21} + 6t_{22} + 9t_{23} + 2t_{31} + 4t_{32} + 6t_{33} = \xi(t), \end{aligned}$$

$$\begin{aligned} 2x_1 + x_2 + 4x_3 \leq 12 &\Leftrightarrow 2(t_{11} + 2t_{12}) + (t_{21} + 2t_{22} + 3t_{23}) + 4(t_{31} + 2t_{32} + 3t_{33}) \leq 12 \\ &\Leftrightarrow 2t_{11} + 4t_{12} + t_{21} + 2t_{22} + 3t_{23} + 4t_{31} + 8t_{32} + 12t_{33} \leq 12, \\ 3x_1 - 2x_2 - x_3 \leq\ 10 &\Leftrightarrow 3(t_{11} + 2t_{12}) - 2(t_{21} + 2t_{22} + 3t_{23}) - (t_{31} + 2t_{32} + 3t_{33}) \leq 10 \\ &\Leftrightarrow 3t_{11} + 6t_{12} - 2t_{21} - 4t_{22} - 6t_{23} - t_{31} - 2t_{32} - 3t_{33} \leq 10, \end{aligned}$$

ce qui donne :

$$d = (-2\ ,\ -4\ ,\ 3\ ,\ 6\ ,\ 9\ ,\ 2\ ,\ 4\ ,\ 6)^T, \quad B = \begin{pmatrix} 2 & 4 & 1 & 2 & 3 & 4 & 8 & 12 \\ 3 & 6 & -2 & -4 & -6 & -1 & -2 & -3 \end{pmatrix}.$$

Le nouveau problème s'écrit alors :

$$\begin{cases} \max & \xi(t) = -2t_{11} - 4t_{12} + 3t_{21} + 6t_{22} + 9t_{23} + 2t_{31} + 4t_{32} + 6t_{33} \\ & 2t_{11} + 4t_{12} + t_{21} + 2t_{22} + 3t_{23} + 4t_{31} + 8t_{32} + 12t_{33} \leq 12, \\ & 3t_{11} + 6t_{12} - 2t_{21} - 4t_{22} - 6t_{23} - t_{31} - 2t_{32} - 3t_{33} \leq 10, \\ & t_{11} + t_{12} \leq 1, \\ & t_{21} + t_{22} + t_{23} \leq 1, \\ & t_{31} + t_{32} + t_{33} \leq 1, \\ & t = (t_{11}\ ,\ t_{12}\ ,\ t_{21}\ ,\ t_{22}\ ,\ t_{23}\ ,\ t_{31}\ ,\ t_{32}\ ,\ t_{33})^T, \quad t_{jk} \in \{0,1\}. \end{cases}$$

3.7.2 Procédure de presolving

Corollaire 3.1. *Dans le problème* (3.43) *de programmation linéaire en nombres entiers à variables bornées, s'il existe $k \in I$ et $r \in J$ satisfaisant :*

$$a_{kj} > 0,\ \forall j \in J, \quad et \quad a_{kr} > b_k, \tag{3.48}$$

alors $x_r^0 = 0$.

Démonstration. Ce corollaire est la conséquence du lemme 3.3. Donc, nous allons appliquer le lemme 3.3 sur le problème reformulé (3.47), soient $k \in I$ et $r \in J$ tels que :

$$\begin{cases} a_{kj} > 0, & \forall j \in J, \\ a_{kr} > b_k, \end{cases}$$

et pour tout $k \in \{1, \ldots, u_j\}$ avec $j \in J$, nous avons :

$$\begin{cases} a_{kj} \times k > 0, & \forall j \in J, \\ a_{kr} \times k > b_k. \end{cases}$$

Selon le lemme 3.3, les variables binaires t_{rk} doivent être égales à zéro ($t_{rk}^0 = 0$, $\forall k \in \{1, \ldots, u_r\}$), ce qui implique que :

$$x_r^0 = \sum_{k=1}^{u_r} (t_{rk}^0 \times k) = 0,$$

correspondant à la valeur optimale de x_r. □

Corollaire 3.2. *Dans le problème* (3.43) *de PLNE, si nous trouvons un indice $r \in J$ qui vérifie l'hypothèse suivante :*

$$a_r \geq 0 \quad et \quad c_r \leq 0, \tag{3.49}$$

alors, la variable $x_r^0 = 0$. Autrement dit, nous pouvons fixer x_r à 0 sans affecter la solution optimale du problème (3.43).

Démonstration. Pour démontrer ce corollaire, nous allons utiliser le lemme 3.4 sur le problème (3.47). D'après l'hypothèse (3.49), et $\forall k \in \{1, \ldots, u_r\}$ avec $r \in J$, nous obtenons :

$$a_r \times k \geq 0 \quad \text{et} \quad c_r \times k \leq 0.$$

Par le lemme 3.4, les variables $t_{rk}^0 = 0$, $\forall k \in \{1, \ldots, u_r\}$. En utilisant l'expression (3.45), nous avons :

$$x_r^0 = \sum_{k=1}^{u_r} (t_{rk}^0 \times k) = 0,$$

ce qui correspond à la valeur optimale de x_r. □

Pour déduire les théorèmes 3.3 et 3.4 dans le problème (3.43) de PLNE à variables bornées, nous supposons que les variables binaires t_{jk} du problème reformulé (3.47) sont fixées à zéro ($t_{jk} = 0$, $\forall j \in J$ et $\forall k \in \{1, \ldots, u_j - 1\}$). Le problème reformulé peut s'écrire de la manière suivante :

$$\begin{cases} \max & \xi(t) = \sum_{j \in J} c_j u_j t_{ju_j} \\ & \sum_{j \in J} a_{ij} u_j t_{ju_j} \leq b_i, \quad \forall i \in I, \\ & t_{ju_j} \in \{0,1\}, \qquad \forall j \in J. \end{cases} \tag{3.50}$$

Ce problème nous permet d'étudier la possibilité de fixer une variable x_r à leur borne supérieure u_r. Autrement dit, si $t^0_{ru_r} = 1$, cela signifie que la variable $x^0_r = u_r$, pour $r \in J$. Le problème (3.50) facilite l'étude de la fixation des variables à leurs bornes supérieures dans le problème (3.43).

Corollaire 3.3. *Si pour $r \in J$, la condition suivante est satisfaite dans le problème* (3.43) *:*

$$a_r \leq 0 \quad et \quad c_r \geq 0, \tag{3.51}$$

alors, la variable x_r sera fixée à u_r.

Démonstration. Pour démontrer ce corollaire, nous appliquons le théorème 3.3 au problème (3.50). En utilisant l'hypothèse (3.51), nous avons :

$$a_r u_r \leq 0 \quad \text{et} \quad c_r u_r \geq 0.$$

En appliquant le théorème 3.3, nous concluons que la variable $t^0_{ru_r} = 1$, ce qui signifie que la valeur optimale de x_r est $x^0_r = u_r$. Par conséquent, lorsque les conditions (3.51) sont satisfaites, la variable x_r est fixée à sa borne supérieure u_r. □

Corollaire 3.4. *Dans le problème* (3.43) *de PLNE à variables bornées, si pour une variable x_r, $r \in J$, on établit les conditions suivantes :*

$$\begin{cases} \max\limits_{j \in J}\{u_j c_j\} = u_r c_r \geq 0, \\ \qquad\qquad et \\ \min\limits_{j \in J}\{u_j a_{ij}\} = u_r a_{ir}, \ avec \ u_r a_{ir} \leq b_i, \ \forall i \in I, \end{cases} \tag{3.52}$$

alors, $x^0_r = u_r$ est une composante optimale.

Démonstration. Dans les conditions (3.52), en posant $c_j u_j = \tilde{c}_j$ et $a_{ij} u_j = \tilde{a}_{ij}$, $\forall i \in I$ et $\forall j \in J$, nous obtenons les conditions suivantes :

$$\begin{cases} \max\limits_{j \in J}\{u_j c_j\} = \max\limits_{j \in J}\{\tilde{c}_j\} = \tilde{c}_r \geq 0, \\ \min\limits_{j \in J}\{u_j a_{ij}\} = \max\limits_{j \in J}\{\tilde{a}_{ij}\} = \tilde{a}_{ir} \leq b_i, \quad \forall i \in I. \end{cases}$$

D'après le théorème 3.4, la variable $t^0_{ru_r} = 1$. Nous concluons que $x^0_r = u_r$, ce qui correspond à sa valeur optimale. Par conséquent, nous pouvons fixer x_r à u_r sans affecter la solution optimale du problème (3.43). □

Exemple 3.4. Nous considérons le problème suivant :

$$\begin{cases} \max & z(x) = c^T x, \\ & Ax \leq b, \\ & 0 \leq x \leq u, \quad x \in \mathbb{N}^6, \end{cases}$$

où

$$A = \begin{pmatrix} 6 & -4 & 6 & 4 & 6 & 5 \\ 8 & -1 & 7 & 1 & 9 & 9 \\ -1 & -2 & 9 & -1 & 3 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 12 \\ 12 \\ 0 \end{pmatrix}, \quad c = \begin{pmatrix} -9 \\ 12 \\ 3 \\ 13 \\ -2 \\ 7 \end{pmatrix}, \quad u = \begin{pmatrix} 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \end{pmatrix}.$$

- Pour $r = 5$:

  $$a_5 = \begin{pmatrix} 6 \\ 9 \\ 3 \end{pmatrix} \geq 0, \quad \text{et} \quad c_5 = -2 \leq 0,$$

  d'après le corollaire 3.2, la variable $x_5^0 = 0$. nous avons

  $$\bar{A} := \begin{pmatrix} 6 & -4 & 6 & 4 & 5 \\ 8 & -1 & 7 & 1 & 9 \\ -1 & -2 & 9 & -1 & 1 \end{pmatrix}, \quad \bar{b} := \begin{pmatrix} 12 \\ 12 \\ 0 \end{pmatrix}, \quad \bar{c} := \begin{pmatrix} -9 \\ 12 \\ 3 \\ 13 \\ 7 \end{pmatrix}, \quad \bar{u} := \begin{pmatrix} 5 \\ 5 \\ 5 \\ 5 \\ 5 \end{pmatrix}, \quad \bar{z} := c_5 x_5^0 = 0.$$

- Pour $r = 2$:

  $$a_2 = \begin{pmatrix} -4 \\ -1 \\ -2 \end{pmatrix} \leq 0, \quad \text{et} \quad c_2 = 12 \geq 0.$$

  D'après le corollaire 3.3, la variable $x_2^0 = 5$. Dans ce cas, nous obtenons

$$\bar{A} := \begin{pmatrix} 6 & 6 & 4 & 5 \\ 8 & 7 & 1 & 9 \\ -1 & 9 & -1 & 1 \end{pmatrix}, \quad \bar{b} := \bar{b} - a_2 x_2^0 = \begin{pmatrix} 32 \\ 17 \\ 10 \end{pmatrix}, \quad \bar{c} := \begin{pmatrix} -9 \\ 3 \\ 13 \\ 7 \end{pmatrix}, \quad \bar{u} := \begin{pmatrix} 5 \\ 5 \\ 5 \\ 5 \end{pmatrix}, \quad \bar{z} := \bar{z} + c_2 x_2^0 = 60.$$

- Pour $r = 4$, nous appliquons le corollaire 3.4 pour le problème réduit, avec $\bar{J} = \{1,3,4,6\}$. Nous avons donc :

  $$\begin{cases} \max\limits_{j \in \bar{J}} \{c_j u_j\} = c_4 u_4 = 65 \geq 0, \\ \min\limits_{j \in \bar{J}} \{a_{ij} u_j\} = a_{i4} u_4, \ \forall i \in I, \qquad \text{avec} \quad a_4 u_4 = (20\ ,\ 5\ ,\ -5)^T \leq \bar{b}. \end{cases}$$

  Par conséquent, $x_4^0 = 5$. Alors, nous obtenons

$$\bar{A} := \begin{pmatrix} 6 & 6 & 5 \\ 8 & 7 & 9 \\ -1 & 9 & 1 \end{pmatrix}, \quad \bar{b} := \bar{b} - a_4 x_4^0 = \begin{pmatrix} 12 \\ 12 \\ 15 \end{pmatrix}, \quad \bar{c} := \begin{pmatrix} -9 \\ 3 \\ 7 \end{pmatrix}, \quad \bar{u} := \begin{pmatrix} 5 \\ 5 \\ 5 \end{pmatrix}, \quad \bar{z} := \bar{z} + c_4 x_4^0 = 125.$$

Nous posons $y = (x_j,\ j \in J_r)$, avec $J_r = \{1,3,6\}$, alors le nouveau problème s'écrit comme suit :

$$\begin{cases} \max & z_r(y) = \bar{c}^T y + 125 \\ & \bar{A}y \leq \bar{b}, \\ & 0 \leq y \leq \bar{u}, \quad y \in \mathbb{N}^3. \end{cases}$$

La solution optimale du problème réduit est $y^0 = (0\ ,\ 0\ ,\ 1)^T$ et $z_r(y^0) = 132$.

Donc, la solution optimale du problème originel est :

$$x^0 = (0\ ,\ 5\ ,\ 0\ ,\ 5\ ,\ 0\ ,\ 1)^T, \quad \text{avec} \quad z(x^0) = z_r(y^0) = 132.$$

Après avoir implémenté sous MATLAB R2021b les deux problèmes ci-dessus l'originel et le réduit, nous avons obtenu la même solution optimale.

3.7.3 Comparaisons numériques

Nous présentons dans cette partie les résultats obtenus lors de nos expérimentations, effectuées sur un ordinateur portable doté d'un processeur I5-4210U avec une fréquence allant de 1.70 à 2.40 GHz et de 12 Go de RAM, et ce, en utilisant le langage de programmation MATLAB version 9.10.0 R2021a. Nous avons analysé les performances de notre technique de réduction de variables pour des problèmes de programmation linéaire en nombres entiers bornés (PLNE) en implémentant les corollaires 3.1,3.2 et 3.3. Pour évaluer l'efficacité de la technique proposée, nous avons comparé le nombre de variables fixées dans deux types de données aléatoires de la matrice A. Nous avons également indiqué les temps d'exécution pour le problème original et le problème réduit en utilisant la fonction 'intlinprog' de MATLAB.

La comparaison est basée sur des problèmes générés aléatoirement, ayant la forme (PLNE) avec les critères suivants :

— Les coefficients c_j sont générés de manière uniforme dans l'intervalle $[-5;5]$;

— Les coefficients u_j sont distribués uniformément dans $[1;5]$;

— Les coefficients b_i sont distribués uniformément dans $[\frac{n}{2};\frac{n}{2}+10]$;

— Les coefficients a_{ij} sont distribués dans les intervalles $[-8;2]$, $[-5;5]$ et $[-2;8]$.

Afin d'évaluer les performances de notre technique, nous adoptons les notations suivantes :

n_1 **:** Le nombre moyen de variables fixées en se basant sur le corollaire 3.1 ;

n_2 **:** Le nombre moyen de variables fixées en se basant sur le corollaire 3.2 ;

n_3 **:** Le nombre moyen de variables fixées en se basant sur le corollaire 3.4 ;

τ_{inf} **:** Taux moyen de variables fixées à la borne inférieure ;

τ_{sup} **:** Taux moyen de variables fixées à la borne supérieure ;

T **:** Le temps d'exécution pour la résolution du problème initial ;

T_p **:** Le temps d'exécution pour la procédure de presolving ;

T_r **:** Le temps d'exécution pour la résolution du problème réduit ;

T_{r+p} **:** Le temps d'exécution pour la procédure de presolving et la résolution du problème réduit.

Afin d'obtenir des résultats fiables et comparables, nous avons généré 100 instances de problème-tests pour chaque dimension. Les résultats obtenus sont représentés dans les tableaux 3.1, 3.2 et 3.3.

n	m	$a_{ij} \in [-5;5]$				$a_{ij} \in [-2;8]$			
		n_1	$\tau_{\text{inf}}\%$	n_2	$\tau_{\text{inf}}\%$	n_1	$\tau_{\text{inf}}\%$	n_2	$\tau_{\text{inf}}\%$
10	3	0.88	17.10	0.79	15.59	3.47	44.26	3.09	39.75
	5	0.29	6.09	0.24	4.42	2.36	28.57	2.51	30.30
20	5	0.54	5.08	0.52	5.11	5.19	32.41	5.11	32.24
	7	0.11	1.03	0.21	2.02	3.30	20.69	3.34	21.08
30	7	0.27	1.76	0.37	2.44	5.33	21.95	5.36	22.08
	10	0.02	0.15	0.02	0.13	3.61	14.93	3.45	14.18
50	10	0.04	0.14	0.04	0.16	5.86	13.97	5.14	12.20
	15	0.01	0.03	0	0	2.51	6.04	2.24	5.36
80	10	0.08	0.19	0.09	0.21	8.92	12.92	8.35	12.07
	20	0	0	0	0	1.68	2.45	1.69	2.49
100	10	0.13	0.25	0.11	0.21	10.72	11.72	10.49	12.02
	20	0	0	0	0	2.07	2.39	1.91	2.21
150	10	0.23	0.29	0.25	0.31	15.56	12.32	16.20	12.2
	20	0	0	0	0	3.31	2.48	3.29	2.48
200	10	0.23	0.22	0.23	0.22	21.2	12.14	21.39	11.99
	20	0	0	0	0	4.14	2.32	4.12	2.31
300	10	0.49	0.30	0.42	0.26	33.23	11.96	32.35	12.01
	20	0	0	0	0	6.26	2.31	6.36	2.35
500	10	0.65	0.23	0.78	0.28	55.07	11.71	54.53	12.03
	20	0	0	0	0	10.72	2.35	10.61	2.33
800	10	1.04	0.23	1.12	0.25	87.10	11.96	86.53	11.87
	20	0	0	0	0	16.71	2.27	16.32	2.22
1000	10	1.48	0.27	1.49	0.27	106.79	11.71	109.63	12.02
	20	0	0	0	0	21.45	2.33	21.58	2.34

TABLEAU 3.1 – L'efficacité du corollaire 3.1 et 3.2

Dans le tableau 3.1, nous avons examiné les corollaires 3.1 et 3.2 pour deux ensembles de données distincts. Pour les problèmes dans lesquels les coefficients $a_{ij} \in [-5;5]$, le nombre moyen de variables fixées par ces corollaires est relativement faible, ce qui indique qu'ils fournissent des résultats assez limités dans ce cas. Cependant, lorsque les coefficients $a_{ij} \in [-2;8]$, les corollaires 3.1 et 3.2 parviennent à fixer un pourcentage significatif de variables pour un nombre limité de contraintes, dépassant 11%. En outre, ces corollaires semblent particulièrement performants lorsque les coefficients sont positifs et que le nombre de contraintes est petit, ce qui permet de fixer un pourcentage acceptable de variables à 0.

n	m	$a_{ij} \in [-8;2]$		$a_{ij} \in [-5;5]$	
		n_3	$\tau_{\text{sup}}\%$	n_3	$\tau_{\text{sup}}\%$
10	3	3.44	82.34	0.86	23.29
	5	2.48	53.57	0.27	6.35
20	5	4.74	48.78	0.44	6.30
	7	3.66	36.24	0.17	2.07
30	7	5.34	35.71	0.24	2.01
	10	3.59	24.14	0.05	0.44
50	10	5.40	21.60	0.06	0.32
	15	2.57	10.71	0	0
80	10	8.42	21.36	0.07	0.23
	20	1.98	5.22	0	0
100	10	10.50	21.70	0.25	0.59
	20	2.05	4.26	0	0
150	10	16.98	23.47	0.23	0.37
	20	2.94	4.11	0	0
200	10	21.47	22.45	0.27	0.33
	20	4.52	4.82	0	0
300	10	33.18	23.11	0.41	0.35
	20	6.78	4.84	0	0
500	10	53.18	22.74	0.68	0.34
	20	11.03	4.70	0	0
800	10	85.78	22.64	1.21	0.38
	20	17.42	4.66	0	0
1000	10	106.70	22.71	1.51	0.37
	20	21.18	4.55	0	0

TABLEAU 3.2 – L'efficacité du corollaire 3.4

Le Tableau 3.2 présente les résultats du corollaire 3.4 pour deux ensembles de données différents. Pour $a_{ij} \in [-8;2]$, ce corollaire fixe un pourcentage élevé de variables, dépassant 21%, pour un nombre limité de contraintes. Tandis que nous observons de moins bonnes performances lorsque le nombre de contraintes est élevé. D'autre part, lorsque les coefficients $a_{ij} \in [-5;5]$, le nombre moyen de variables fixées est relativement faible, indiquant des performances limitées même lorsque le nombre de contraintes est petit. En outre, les résultats indiquent que ce corollaire est plus performant lorsque les coefficients sont négatifs et que le nombre de contraintes est petit, permettant de fixer un pourcentage élevé de variables à leurs bornes supérieures.

n	m	T	T_p	T_r	T_{p+r}
10	3	0.0291	0.0003	0.0206	0.0209
	5	0.0223	0.0003	0.0202	0.0205
20	5	0.0303	0.0004	0.0277	0.0281
	7	0.0211	0.0003	0.0175	0.0178
30	7	0.0234	0.0003	0.0209	0.0212
	10	0.0206	0.0003	0.0184	0.0187
50	10	0.0223	0.0003	0.0183	0.0186
	15	0.0219	0.0003	0.0173	0.0176
80	10	0.0220	0.0003	0.0155	0.0158
	20	0.0242	0.0003	0.0189	0.0192
100	10	0.0205	0.0003	0.0191	0.0194
	20	0.0225	0.0003	0.0184	0.0187
150	10	0.0233	0.0003	0.0180	0.0183
	20	0.0233	0.0003	0.0214	0.0217
200	10	0.0263	0.0004	0.0233	0.0237
	20	0.0244	0.0003	0.0220	0.0223
300	10	0.0247	0.0003	0.0222	0.0225
	20	0.0280	0.0004	0.0206	0.0210
500	10	0.0267	0.0004	0.0222	0.0226
	20	0.0278	0.0005	0.0239	0.0244
800	10	0.0292	0.0005	0.0255	0.0260
	20	0.0331	0.0005	0.0281	0.0286
1000	10	0.0284	0.0006	0.0250	0.0256
	20	0.0305	0.0005	0.0288	0.0293

TABLEAU 3.3 – Comparaison des temps d'exécution entre (P) et (P_r)

Le tableau 3.3 représente une comparaison des temps d'exécution entre les problèmes originels et les problèmes réduits avec l'ajout de la procédure de presolving basée sur le corollaire 3.4 et pour laquelle les coefficients a_{ij} sont contraints à l'intervalle $[-8;2]$. Les résultats obtenus montrent que cette procédure est très prometteuse et offre de bonnes performances.

3.8 Conclusion

Ce chapitre a exploré quelques techniques de presolving appliquées aux problèmes de programmation linéaire en nombres entiers (PLNE). Nous avons commencé par introduire le concept de problème réduit, qui permet de simplifier les problèmes tout en préservant leurs propriétés fondamentales. Ensuite, plusieurs méthodes de presolving ont été présentées, notamment celles axées sur les contraintes, les bornes des variables, et les particularités des PLNE à variables spécifiques (non négatives, binaires, ou bornées).

Les résultats obtenus montrent que le presolving joue un rôle important dans la simplification des problèmes. Il permet de réduire la taille des problèmes en éliminant des variables, des contraintes redondantes et en diminuant le temps de calcul nécessaire pour trouver une solution. Ces contributions posent

des bases solides pour intégrer davantage les différentes techniques de presolving dans des approches heuristiques ou des solveurs industriels.

Cependant, certaines limites ont également été identifiées, en particulier lorsque les coefficients des matrices de contraintes prennent des valeurs à la fois positives et négatives. Ces observations ouvrent la voie à des améliorations et généralisations futures des techniques de presolving.

Chapitre 4

Méthode heuristique d'arrondissement et presolving

Sommaire

4.1 Introduction

La résolution de problèmes de Programmation Linéaire en Nombres Entiers (PLNE) de grande taille peut souvent être exigeante en termes de calculs et de temps. Face à ces défis, les méthodes heuristiques offrent une alternative viable, capable de fournir des solutions de bonne qualité en un temps considérablement réduit grâce à certaines approximations.

La recherche sur les méthodes heuristiques pour résoudre les problèmes de programmation linéaire en nombres entiers (PLNE) a connu des avancées significatives durant ces vingt dernières années. Berthold [12] a exploré diverses techniques d'arrondi, jetant ainsi les bases de nombreuses innovations dans ce domaine. Les principaux travaux liés à celui-ci incluent Bertacco et al. [11], Melo et al. [57], Hendel [43], Achterberg et al. [2], Berthold [13], Ding et al. [28], Shen et al. [66], Witzig et Gleixner [69], ainsi que Halbig et al. [42].

Fischetti et al. [30] ont introduit des innovations dans la recherche de proximité pour les problèmes mixtes en 0-1, tandis que Bibi et al. [18, 22] ont proposé une méthode basée sur une approche heuristique d'arrondi pour résoudre les problèmes de programmation linéaire en nombres entiers et mixtes. Quant à Berthold [14], il a décrit les méthodes heuristiques appliquées à la programmation non linéaire mixte en nombres entiers.

Ce chapitre présente une méthode heuristique basée sur l'arrondi, qui intègre des techniques de presolving issues de [64], pour trouver des solutions réalisables aux problèmes de PLNE à variables bornées, avec une matrice des contraintes positive et un second membre positif. L'approche proposée repose sur deux étapes principales : une réduction des bornes des variables, suivie d'un processus d'arrondi guidé par un problème auxiliaire obtenu à partir du problème initial. Cette stratégie tire parti des informations de la relaxation continue du problème pour guider efficacement le processus d'arrondi et améliorer la qualité des solutions.

La méthode heuristique que nous proposons présente des similitudes avec celles décrites dans [12], [13], [11] et [43], notamment en ce qui concerne l'objectif commun et l'utilisation de l'arrondi comme opération de base. Cependant, notre heuristique se distingue par sa spécificité à une classe particulière de problèmes, alors que les autres méthodes heuristiques sont plus générales. Contrairement à ces dernières, notre heuristique garantit une solution réalisable dès l'arrondi initial. De plus, la nouveauté de notre approche réside dans la construction d'un problème auxiliaire. Enfin, notre méthode utilise le coût unitaire comme principal critère pour sélectionner les variables à ajuster, tandis que les autres méthodes se basent souvent sur la proximité des valeurs entières ou d'autres métriques.

Les performances de cette heuristique sont évaluées à travers des expériences numériques, qui démontrent son efficacité pour produire des solutions réalisables de qualité, tout en réduisant considérablement les temps de calcul.

4.2 Position du problème

Nous considérons le problème de programmation linéaire à variables entières bornées suivant :

$$\begin{cases} \max\ z(x) = c^T x, \\ \quad Ax \leq b, \\ \quad 0 \leq x \leq u, \quad x \in \mathbb{N}^n, \end{cases} \tag{4.1}$$

où la matrice $A \in \mathscr{M}_{m\times n}(\mathbb{R}_+)$ est une matrice des contraintes d'ordre $m \times n$, et à coefficients positifs ou nuls, $b \in \mathbb{R}_+^m$, $u \in \mathbb{R}_+^n$, et $c \in \mathbb{R}_+^n$, $x \in \mathbb{N}^n$.

L'ensemble des solutions réalisables du problème (4.1) sera désigné par S^+ :

$$S^+ = \{x \in \mathbb{N}^n \ :\ Ax \leq b,\ 0 \leq x \leq u\}.$$

Enfin, la relaxation continue du problème de PLNE est le problème de programmation linéaire obtenu en supprimant les contraintes $x \in \mathbb{N}^n$ dans le problème (4.1). L'ensemble des solutions réalisables sera noté par :

$$S^R = \{x \in \mathbb{R}^n \ :\ Ax \leq b,\ 0 \leq x \leq u\}.$$

4.3 Méthode heuristique d'arrondissement

La méthode heuristique proposée, comme indiqué précédemment, se décompose en deux étapes principales. La première étape implique une réduction des bornes, suivie d'un processus d'arrondissement appliqué à un problème auxiliaire, dérivé du problème initial. Les détails de ces étapes sont exposés ci-dessous.

4.3.1 Réduction des bornes

La réduction des bornes commence avec la solution optimale du problème relaxé de PLNE. Soit x^R une solution optimale du problème relaxé. Pour chaque indice $j \in J$, on décompose x_j^R comme suit :

$$x_j^R = \lfloor x_j^R \rfloor + f_j,\ 0 \leq f_j < 1, \tag{4.2}$$

où $\lfloor x_j^R \rfloor$ représente la partie entière de x_j^R et $f_j = x_j^R - \lfloor x_j^R \rfloor$ est sa partie fractionnaire.

Nous définissons ensuite deux ensembles d'indices basés sur cette décomposition :

$$J_I = \{j \in J \ :\ f_j = 0\}, \quad J_F = \{j \in J \ :\ 0 < f_j < 1\}, \tag{4.3}$$

où l'ensemble J_I contient les indices des variables dont les valeurs sont déjà des entiers dans la solution optimale relaxée, et J_F contient les indices des variables fractionnaires dans la solution optimale x^R.

Ces deux ensembles sont disjoints et couvrent l'ensemble des indices J, soit :

$$J_I \cap J_F = \emptyset, \quad J_I \cup J_F = J.$$

La réduction des bornes des variables se fait comme suit :

$$\bar{l}_j \leq x_j \leq \bar{u}_j, \quad j \in J, \tag{4.4}$$

où $\bar{l}_j = \lfloor x_j^R \rfloor$, $\bar{u}_j = \min\{\lfloor u_j \rfloor, \lfloor x_j^R \rfloor + 1\}$.

Remarque 4.1. Dans certains cas, cette réduction des bornes peut entraîner la fixation de certaines variables. Cela se produit lorsque les nouvelles bornes inférieure ($\bar{l}_j$) et supérieure ($\bar{u}_j$) coïncident avec la partie entière de la solution relaxée, c'est-à-dire :

$$\bar{l}_j = \bar{u}_j, \quad j \in J_F \quad \Leftrightarrow \quad \lfloor u_j \rfloor = \lfloor x_j^R \rfloor, \quad j \in J. \tag{4.5}$$

Dans ce cas, on définit l'ensemble J_u comme suit :

$$J_u = \left\{ j \in J \ : \ \lfloor u_j \rfloor = \lfloor x_j^R \rfloor \right\}. \tag{4.6}$$

Les variables x_j, $j \in J_u$, sont alors fixées à la partie entière inférieure ($x_j^a = \lfloor x_j^R \rfloor$) et sont exclues de l'étape suivante. Lensemble des indices restants est donné par :

$$\bar{J} = J \setminus J_u, \quad |\bar{J}| = n_r \leq n. \tag{4.7}$$

4.3.2 Problème auxiliaire

La seconde étape de l'heuristique s'appuie sur un problème auxiliaire dérivé du problème initial de PLNE. Ce problème utilise les informations issues de la solution optimale de la relaxation continue du PLNE afin d'orienter le processus d'arrondissement.

4.3.2.1 Variables du problème auxiliaire

Soit x^R la solution optimale de la relaxation continue du PLNE. Pour chaque variable x_j^R, $j \in \bar{J}$, nous définissons une nouvelle variable y_j représentant la partie fractionnaire de x_j^R :

$$y_j = x_j^R - \lfloor x_j^R \rfloor, \quad 0 \leq y_j \leq 1, \quad j \in \bar{J}. \tag{4.8}$$

L'objectif du problème auxiliaire est de transformer ces variables en des variables binaires $y_j \in \{0;1\}$. Cette contrainte binaire impose que la composante x_j^R soit arrondie à sa partie entière inférieure ($\lfloor x_j^R \rfloor$) lorsque $y_j = 0$, soit à $\lfloor x_j^R \rfloor + 1$ lorsque $y_j = 1$.

4.3.2.2 Fonction Objectif

La fonction objectif du problème de PLNB est directement issue de celle du problème initial, adaptée pour refléter les nouvelles contraintes :

$$\begin{aligned} z(x) = \sum_{j\in J} c_j x_j = \sum_{j\in J} c_j (y_j + \lfloor x_j^R \rfloor) &= \sum_{j\in J} c_j y_j + \sum_{j\in J} c_j \lfloor x_j^R \rfloor \\ &= \sum_{j\in J\setminus J_u} c_j y_j + \cancel{\sum_{j\in J_u} c_j y_j} + \sum_{j\in J} c_j \lfloor x_j^R \rfloor \\ &= \sum_{j\in \bar{J}} c_j y_j + \underline{z}_R, \end{aligned} \tag{4.9}$$

où $\underline{z}_R = \sum_{j\in J} c_j \lfloor x_j^R \rfloor$, représentant la contribution des parties entières inférieures des variables dans la solution relaxée et $y_j = 0$, pour tout $j \in J_u$.

Ainsi, la fonction objectif du problème auxiliaire devient :

$$\mathfrak{z}(y) = \bar{c}^T y + \underline{z}_R, \tag{4.10}$$

où $\bar{c} = c(\bar{J}) = (c_j,\ j \in \bar{J})$.

4.3.2.3 Contraintes du Problème Auxiliaire

Les contraintes du problème auxiliaire sont dérivées des contraintes du problème de PLNE initial en remplaçant x_j par $y_j + \lfloor x_j^R \rfloor$. On a :

$$\sum_{j\in J} a_j x_j \leq b,$$

on obtient :

$$\begin{aligned} \sum_{j\in J} a_j (y_j + \lfloor x_j^R \rfloor) \leq b \quad &\Leftrightarrow \quad \sum_{j\in J\setminus J_u} a_j y_j + \cancel{\sum_{j\in J_u} a_j y_j} + \sum_{j\in J} a_j \lfloor x_j^R \rfloor \leq b \\ &\Leftrightarrow \quad \sum_{j\in \bar{J}} a_j y_j \leq b - \sum_{j\in J} a_j \lfloor x_j^R \rfloor. \end{aligned}$$

En posant $\bar{b} = b - \sum_{j\in J} a_j \lfloor x_j^R \rfloor$, les contraintes du problème auxiliaire deviennent :

$$\sum_{j\in \bar{J}} a_j y_j \leq \bar{b} \quad \Leftrightarrow \quad \bar{A} y \leq \bar{b}. \tag{4.11}$$

où $\bar{A} = A(I,\bar{J}) = (a_j,\ j \in \bar{J})$.

Les nouvelles variables y_j sont également des variables binaires :

$$y = (y_j,\ j \in \bar{J}), \quad y_j \in \{0,1\}. \tag{4.12}$$

4.3.2.4 Formulation Finale du Problème auxiliaire

Le problème auxiliaire PLNB se présente sous la forme suivante :

$$\begin{cases} \max & \mathfrak{z}(y) = \bar{c}^T y + \underline{z}_R, \\ & \bar{A}y \leq \bar{b}, \\ & y \in \{0,1\}^{n_r}. \end{cases} \tag{4.13}$$

La résolution du problème (4.13) apporte des indications précises sur la façon d'arrondir les variables de la solution relaxée du problème (4.1). En imposant que les variables y_j soient binaires, le problème auxiliaire oriente l'arrondissement des variables x_j^R vers leurs valeurs entières, tout en respectant les contraintes initiales. Cette étape est essentielle pour assurer que la solution obtenue soit réalisable et suboptimale.

4.3.3 Principe de la méthode heuristique

Le principe de l'heuristique proposée repose sur les étapes suivantes : Après avoir résolu la relaxation continue du problème de PLNE et arrondi la solution à la partie entière inférieure, toutes les variables associées à l'ensemble J_u sont exclues du problème auxiliaire et, par conséquent, du processus d'arrondissement. Cette élimination réduit le nombre de variables à considérer et permet à l'heuristique de se concentrer sur celles nécessitant un arrondi pour obtenir des valeurs entières, ce qui peut accélérer significativement la convergence de la solution relaxée vers une solution entière.

Précisément, l'heuristique se concentre sur les variables ayant le plus grand potentiel d'impact sur la valeur de la fonction objectif. Cela est réalisé en sélectionnant les variables associées aux coûtx unitaires les plus élevés, tout en respectant les contraintes. Soit j_0 l'indice défini par :

$$j_0 = \arg\max_{j \in \bar{J}} \{c_j\}. \tag{4.14}$$

Cet indice correspond à la variable présentant le coût le plus élevé dans la fonction objectif parmi les variables non encore arrondies. On vérifie si les contraintes permettent de fixer la variable $y_{j_0} = 1$. En d'autres termes, on s'assure que la condition suivante est satisfaite $a_{j_0} \leq \bar{b}$, à savoir que la variable peut être arrondie vers sa partie entière supérieure ($y_{j_0} = 1$), ce qui correspond à $x_{j_0}^R$ arrondi à sa borne supérieure entière. Sinon, on fixe $y_{j_0} = 0$, ce qui implique un arrondi de $x_{j_0}^R$ vers sa partie entière inférieure. Après avoir fixé la variable y_{j_0}, le vecteur $\bar{b}$ est mis à jour comme suit :

$$\bar{b} := \bar{b} - a_{j_0} y_{j_0}. \tag{4.15}$$

Par ailleurs, l'indice j_0 est supprimé de l'ensemble $\bar{J}$, réduisant ainsi l'ensemble des variables à traiter.

Propriété 4.1. *La solution entière construite à l'aide de ce principe est nécessairement réalisable, car chaque variable est arrondie en respectant les contraintes du problème.*

Démonstration. Considérons x^R une solution optimale relaxée et x^a une solution arrondie obtenue par la procédure ci-dessus.

1. Le vecteur x^a défini par :

   $$x_j^a = \lfloor x_j^R \rfloor, \quad \forall j \in J,$$

   est une solution réalisable, car :

   $$\sum_{j\in J} a_j x_j^a = \sum_{j\in J} a_j \lfloor x_j^R \rfloor \leq \sum_{j\in J} a_j x_j^R \leq b.$$

   Ainsi, x^a satisfait les contraintes du problème, vu que par construction on a :

   $$0 \leq \lfloor x_j^R \rfloor = x_j^a \leq x_j^R \leq u_j, \quad j \in J.$$

   La solution $x^a = \lfloor x^R \rfloor$ est une solution entière réalisable.

2. Supposons qu'il existe une solution entière réalisable $\hat{x}^a$, obtenue en arrondissant x^R. L'objectif est de montrer que si un indice $j_0 \in \bar{J}$ vérifie $\hat{x}_{j_0}^a = \lfloor x_{j_0}^R \rfloor$ et les relations (4.14) et $a_{j_0} \leq \bar{b}$, où $\bar{b} = b - \sum_{j\in J} a_j \hat{x}_j^a$, alors on peut construire une solution $\tilde{x}^a$ encore meilleure.

   Pour un $j_0 \in \bar{J}$ tel que $a_{j_0} \leq \bar{b}$, définissons :

   $$\tilde{x}_j^a = \begin{cases} \hat{x}_j^a, & j \in \bar{J} \setminus \{j_0\} \cup J_u, \\ \hat{x}_{j_0}^a + 1, & j = j_0. \end{cases}$$

   Puisque $a_{j_0} \leq \bar{b}$, alors on a :

   $$\begin{aligned} a_{j_0} \leq \bar{b} \quad &\Leftrightarrow \quad a_{j_0} \leq b - \sum_{j\in J} a_j \hat{x}_j^a \\ &\Leftrightarrow \quad a_{j_0} + \sum_{j\in J} a_j \hat{x}_j^a \leq b \\ &\Leftrightarrow \quad a_{j_0} + a_{j_0} \hat{x}_{j_0}^a + \sum_{j\in J\setminus\{j_0\}} a_j \hat{x}_j^a \leq b \\ &\Leftrightarrow \quad a_{j_0} \left(\hat{x}_{j_0}^a + 1\right) + \sum_{j\in J\setminus\{j_0\}} a_j \hat{x}_j^a \leq b \\ &\Leftrightarrow \quad a_{j_0} \tilde{x}_{j_0}^a + \sum_{j\in J\setminus\{j_0\}} a_j \tilde{x}_j^a \leq b \\ &\Leftrightarrow \quad \sum_{j\in J} a_j \tilde{x}_j^a \leq b. \end{aligned}$$

   Ainsi, le vecteur $\tilde{x}^a$ est réalisable. De plus, on a :

   $$\begin{aligned} z(\hat{x}^a) = c^T \hat{x}^a = \sum_{j\in J} c_j^T \hat{x}_j^a &= \sum_{j\in J\setminus\{j_0\}} c_j^T \hat{x}_j^a + c_{j_0}^T \hat{x}_{j_0}^a \\ &< \sum_{j\in J\setminus\{j_0\}} c_j^T \hat{x}_j^a + c_{j_0}^T \left(\hat{x}_{j_0}^a + 1\right) = \sum_{j\in J\setminus\{j_0\}} c_j^T \tilde{x}_j^a + c_{j_0}^T \tilde{x}_{j_0}^a = \sum_{j\in J} c_j^T \tilde{x}_j^a = z(\tilde{x}^a). \end{aligned}$$

Ce processus garantit qu'une solution entière réalisable peut être construite à partir de x^R, avec une valeur objectif potentiellement améliorée à chaque itération. □

4.3.4 Procédure de presolving

Dans le chapitre 3, nous avons introduit et détaillé différentes techniques de presolving. En particulier, le tableau (3.1) montre que les lemmes 3.2 et 3.3 sont particulièrement efficaces lorsque les coefficients de la matrice des contraintes sont positifs.

Dans ce chapitre, nous appliquons la technique décrite dans le lemme 3.2 pour supprimer l'indice r. Cette approche permet d'accélérer significativement la procédure de la méthode heuristique. Plus précisément, pour $k \in I$ et $r \in J \setminus J_u$, si :

$$a_{kr} > \bar{b}_k, \tag{4.16}$$

alors $y_r = 0$, et l'indice r est retiré de l'ensemble $J \setminus J_u$.

Ci-dessous, un pseudo-code de l'heuristique illustre les différentes étapes décrites :

Algorithme 8 : Méthode heuristique d'arrondi

Entrées : A, b, c, J, x^R
Sorties : x^a, z_a

`Initialiser` J_I, J_F, J_u
pour $j \in J_F$ **faire**
  $x^a_j \leftarrow \lfloor x^R_j \rfloor$
fin pour
$\bar{b} \leftarrow b - Ax^a$
$\bar{J} \leftarrow J \setminus J_u$
tant que $\bar{J} \neq \emptyset$ **faire**
  $j_0 = \arg\max_{j \in \bar{J}} \{c_j\}$
  si $a_{j_0} \leq \bar{b}$ **alors**
    $x^a_{j_0} \leftarrow x^a_{j_0} + 1$
    $\bar{b} \leftarrow \bar{b} - a_{j_0}$
  fin si
  $\bar{J} \leftarrow \bar{J} \setminus \{j_0\}$
  $R = \{j \in \bar{J} : \exists k \in I, \ a_{kr} > \bar{b}_k\}$
  $\bar{J} \leftarrow \bar{J} \setminus \{R\}$
fin tq
$z_a \leftarrow c^T x^a$
retourne x^a, z_a

4.4 Exemple illustratif

L'exemple suivant illustre le fonctionnement de l'heuristique proposée. Considérons le problème de programmation linéaire en nombres entiers suivant (P) :

$$(\mathrm{P}) \begin{cases} \max \ z(x) = c^T x, \\ \quad Ax \leq b, \\ \quad 0 \leq x \leq u, \quad x \in \mathbb{N}^n, \end{cases}$$

avec les données suivantes :

$$A = \begin{pmatrix} 5 & 2 & 1 & 1 & 2 \\ 1 & 1 & 6 & 8 & 2 \\ 5 & 2 & 5 & 2 & 6 \end{pmatrix}, \quad b = \begin{pmatrix} 15.12 \\ 52.03 \\ 30.29 \end{pmatrix}, \quad c = \begin{pmatrix} 5 \\ 2 \\ 7 \\ 6 \\ 6 \end{pmatrix}, \quad u = \begin{pmatrix} 8.8 \\ 8.55 \\ 4.13 \\ 4.19 \\ 6.98 \end{pmatrix}.$$

En résolvant le problème relaxé, nous obtenons la solution suivante :

$$x^R = \begin{pmatrix} 1.5564 \\ 0 \\ 2.8256 \\ 4.19 \\ 0 \end{pmatrix} \quad \text{avec} \quad z(x^R) = 52.7012.$$

Étant donné que cette solution n'est pas entière, nous appliquons l'algorithme pour trouver le meilleur arrondi.

- **Initialisation**
  - Ensembles d'indices :

    $$J_I = \{2,5\}, \quad J_F = \{1,3,4\}, \quad J_u = \{4\}.$$

  - La solution initiale par partie entière inférieure :

    $$x^a = \lfloor x^R \rfloor = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 4 \\ 0 \end{pmatrix}, \quad \text{avec} \quad z_a = c^T x^a = 43.$$

  - La mise à jour du second membre :

    $$\bar{b} = b - Ax^a = \begin{pmatrix} 4.12 \\ 7.03 \\ 7.29 \end{pmatrix}.$$

  - L'ensemble réduit des variables à traiter :

    $$\bar{J} := J \setminus J_u = \{1,2,3,5\}.$$

  - Le problème auxiliaire sera :

    $$\begin{cases} \max & \mathfrak{z}(y) = \bar{c}^T y + \underline{z}_R, \\ & \bar{A}y \leq \bar{b}, \\ & y \in \{0,1\}^4 \end{cases}$$

    où

    $$\bar{A} = \begin{pmatrix} 5 & 2 & 1 & 2 \\ 1 & 1 & 6 & 2 \\ 5 & 2 & 5 & 6 \end{pmatrix}, \quad \bar{b} = \begin{pmatrix} 4.12 \\ 7.03 \\ 7.29 \end{pmatrix}, \quad \bar{c} = \begin{pmatrix} 5 \\ 2 \\ 7 \\ 6 \end{pmatrix}.$$

- **Itérations de l'Algorithme**

  - Première itération

    Nous avons $a_{11} = 5 > \bar{b}_1 = 4.12$, d'après le corollaire 3.1 $y_1 = 0$, et l'ensemble $\bar{J} := \bar{J} \setminus \{1\} = \{2,3,5\}$.

    $$j_0 = \arg\max_{j \in \bar{J}} \{c_j\} = 3.$$

    Puisque $a_3 \leq \bar{b}$, on fixe $x_3^a := x_3^a + 1$. Mise à jour de l'ensemble $\bar{J} := \bar{J} \setminus \{j_0\} = \{2,5\}$, ce qui donne

    $$\bar{A} = \begin{pmatrix} 2 & 2 \\ 1 & 2 \\ 2 & 6 \end{pmatrix}, \quad \bar{b} := \bar{b} - a_3 = \begin{pmatrix} 3.12 \\ 1.03 \\ 2.29 \end{pmatrix}, \quad \bar{c} = \begin{pmatrix} 2 \\ 6 \end{pmatrix}, \quad z_a := z_a + c_3 = 50.$$

  - Deuxième itération

    Nous avons $a_{53} = 6 > \bar{b}_3 = 2.29$, d'après le corollaire 3.1 $y_5 = 0$, et l'ensemble $\bar{J} := \bar{J} \setminus \{5\} = \{2\}$. Alors nous choisissons $j_0 = 2$. Puisque $a_2 \leq \bar{b}$, on fixe $x_2^a := x_2^a + 1$, ce qui donne $z_a := z_a + c_2 = 52$, $\bar{J} := \emptyset$. L'ensemble $\bar{J}$ est maintenant vide, et l'algorithme s'arrête ici.

La solution entière obtenue après application de l'algorithme est :

$$x^a = \begin{pmatrix} 1 \\ 1 \\ 3 \\ 4 \\ 0 \end{pmatrix}, \quad \text{avec} \quad z_a = z(x^a) = 52.$$

Puisque $z(x^a) < z(x^R) < z(x^a) + 1$, et que $x^a \in \mathbb{N}^5$, $c \in \mathbb{N}^5$, le vecteur x^a est la solution optimale du problème initial.

4.5 Résultats numériques

Dans cette section, nous présentons les résultats numériques mettant en évidence les performances de l'heuristique proposée. Les tests ont été réalisés sur un ordinateur portable équipé d'un processeur Intel I7-10750H cadencé à 2,60 GHz et doté de 24 Go de RAM. L'implémentation de notre heuristique a été effectuée à l'aide de Matlab R2023b.

Le temps d'exécution et la précision des solutions obtenues avec notre méthode sont comparés à ceux obtenus en utilisant le solveur Gurobi sur des problèmes de programmation linéaire en nombres entiers générés aléatoirement, selon la procédure suivante : les coefficients (a_{ij}) sont générés aléatoirement suivant une distribution uniforme dans l'intervalle $[0;100]$; les coefficients (c_j) sont calculés selon la formule :

$$c_j = \left\lfloor \frac{1}{m} \sum_{i \in I} a_{ij} + \frac{5}{1+\eta} \right\rfloor,$$

où η est une variable aléatoire uniformément distribuée sur l'intervalle $[0;1]$. Quant aux coefficients (b_i), ils sont générés selon la formule :

$$b_i = \beta \times \sum_{j \in J} a_{ij},$$

où β est une variable aléatoire générée dans l'intervalle $[0;1]$. Enfin, les bornes supérieures $\left(u_j\right)$ sont générées uniformément dans l'intervalle $[2;12]$.

Pour évaluer la performance de notre méthode par rapport au solveur Gurobi, nous utilisons les mesures suivantes :

- T_1 : le temps d'exécution avec le solveur Gurobi ;
- T_2 : le temps d'exécution pour résoudre le problème relaxé et le temps d'exécution concernant notre algorithme ;
- $\Delta z = \dfrac{z^* - z_a}{z^*}$: la différence relative entre la valeur optimale de la fonction objectif obtenue par Gurobi (z^*) et celle obtenue par notre méthode (z_a) ;
- τ : le taux moyen des variables arrondies dont les valeurs coïncident exactement avec celles obtenues par Gurobi.

n	m	T_1	T_2	Δz	$\tau\%$
50	15	0.0725	0.0289	0.0272	95.74
	25	0.0734	0.0290	0.0222	96.38
100	25	0.3771	0.0344	0.0286	96.35
	35	0.2670	0.0341	0.0329	96.69
150	35	0.3463	0.0426	0.0284	96.76
	50	0.2397	0.0393	0.0324	97.37
200	50	0.7217	0.0407	0.0291	97.80
	70	0.6261	0.0434	0.0345	98.02
300	50	0.6428	0.0446	0.0175	98.42
	70	0.4815	0.0400	0.0292	98.33
400	50	0.9581	0.0418	0.0190	98.44
	70	0.6181	0.0440	0.0279	98.60
500	50	1.4129	0.0464	0.0188	98.59
	70	2.3085	0.0685	0.0261	98.54

TABLEAU 4.1 – Résultats numériques.

Les résultats obtenus, présentés dans le tableau 4.1, montrent clairement que le temps d'exécution pour résoudre le problème relaxé, puis notre algorithme, est inférieur à celui du solveur Gurobi pour tous les problèmes testés. Concernant la qualité des solutions, on observe que la différence relative Δz reste constamment inférieure à 0,035. De plus, le taux de fixation des variables à leurs valeurs optimales dépasse 95% pour tous les problèmes testés. Cela indique que notre méthode parvient à fixer un grand nombre de variables à leurs valeurs optimales dès les premières étapes de l'algorithme, contribuant ainsi à son efficacité.

4.6 Conclusion

Dans ce chapitre, nous avons présenté une heuristique combinant la réduction des bornes et l'arrondissement pour générer des solutions entières réalisables pour des problèmes de programmation linéaire en nombres entiers, avec une matrice de contraintes et un second membre positifs. L'étape d'arrondissement repose sur un problème auxiliaire formulé comme un programme linéaire binaire.

Les résultats expérimentaux ont montré que la méthode proposée offre un compromis intéressant entre rapidité d'exécution et qualité des solutions, s'avérant particulièrement efficace pour les problèmes de grande taille.

Conclusion générale

Cette thèse s'inscrit dans le domaine de l'optimisation combinatoire, en se concentrant sur l'amélioration des performances des solveurs pour les problèmes de programmation linéaire en nombres entiers (PLNE). Plus précisément, nous avons étudié et développé des techniques de presolving visant à réduire la taille des problèmes, ainsi qu'une méthode heuristique pour obtenir des solutions suboptimales de bonne qualité dans des temps de calcul réduits.

Dans le cadre du presolving, nous avons proposé une procédure permettant de fixer certaines variables à leurs bornes (inférieure ou supérieure) sans affecter l'optimalité de la solution. Cette méthode s'est révélée particulièrement efficace pour simplifier les problèmes et réduire significativement le nombre de variables à considérer. Les expérimentations ont démontré que cette approche offre de bonnes performances en termes de temps d'exécution après avoir réduit efficacement la taille des problèmes, au moyen du presolving.

Par ailleurs, une méthode a été introduite pour trouver des solutions entières réalisables à partir de la solution optimale du problème relaxé dans le cas où les coefficients de la matrice des contraintes sont positifs ou nuls. Cette approche a montré qu'elle offrait un compromis intéressant entre rapidité d'exécution et qualité des solutions, en particulier pour les problèmes de grande taille. Les résultats expérimentaux ont confirmé que la méthode produit des solutions suboptimales de bonne qualité dans des temps raisonnables.

Malgré les résultats encourageants, certaines limites ont été relevées :

- Les techniques de presolving proposées montrent des performances réduites lorsque les coefficients de la matrice des contraintes prennent des valeurs non nécessairement négatives ou positves, comme dans l'intervalle $[-5;5]$;
- L'heuristique d'arrondissement fonctionne principalement pour les matrices de contraintes positives, ce qui limite son domaine d'application ;
- Bien que les solutions obtenues par la méthode heuristique soient de bonne qualité, elles demeurent suboptimales. Une analyse approfondie est nécessaire pour améliorer encore leur précision et leur proximité avec la solution optimale.

Les travaux réalisés dans cette thèse ouvrent des perspectives intéressantes pour développer et généraliser les approches proposées, en particulier pour les axes suivants :

1. Développement de procédures de presolving pour les problèmes de programmation non linéaire en nombres entiers ;
2. Conception d'une méthode heuristique adaptée aux matrices de contraintes, avec des coefficients non nécessairement positifs ;
3. Proposition d'une estimation pour tester la suboptimalité des solutions ;
4. Explorer des méthodes hybrides combinant heuristiques et algorithmes exacts pour trouver des solutions optimales ou suboptimales.

En conclusion, cette thèse contribue au domaine de l'optimisation combinatoire, en proposant des techniques de presolving et des heuristiques adaptées aux problèmes de PLNE à variables bornées, où auparavant sauf les variables bornées inférieurement par zéro ont été traités par les différents auteurs. Ces travaux constituent une base solide pour des recherches futures, visant à améliorer les performances des solveurs et à répondre aux défis posés par des problèmes d'optimisation de plus grande taille et à structure plus complexe.

Bibliographie

[1] T Achterberg. *Constraint Integer Programming*. PhD thesis, Technical University of Berlin, Berlin, July 2007.

[2] T. Achterberg, T. Berthold, and G. Hendel. Rounding and propagation heuristics for mixed integer programming. In *Operations Research Proceedings 2011*, pages 71–76. Springer Berlin Heidelberg, 2012.

[3] T. Achterberg, R. E. Bixby, Z. Gu, E Rothberg, and D. Weninger. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, 32(2):473–506, 2020.

[4] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Mathematical Programming*, 71(2):221–245, 1995.

[5] D. A. Babayev and S. S. Mardanov. Reducing the number of variables in integer and linear programming problems. *Computational Optimization and Applications*, 3(2):99–109, 1994.

[6] E. M. Beale. Branch and bound methods for mathematical programming systems, 1979.

[7] M. Bentobache. *Sur les méthodes mathématiques de la programmation linéaire et quadratique*. Thèse de doctorat, Université de Béjaia, Béjaia, Février 2013.

[8] M. Bentobache and M. O. Bibi. Adaptive method with hybrid direction : theory and numerical experiments. *Proceedings of the Optimization*, pages 24–27, 2011.

[9] M. Bentobache and M. O. Bibi. A two-phase support method for solving linear programs : Numerical experiments. *Mathematical Problems in Engineering*, 2012(1):482193, 2012.

[10] M. Bentobache and M. O. Bibi. A hybrid direction algorithm with long step rule for linear programming : numerical experiments. In *Modelling, Computation and Optimization in Information Systems and Management Sciences : Proceedings of the 3rd International Conference on Modelling, Computation and Optimization in Information Systems and Management Sciences-MCO 2015-Part I*, pages 333–344. Springer, 2015.

[11] L. Bertacco, M. Fischetti, and A. Lodi. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization*, 4(1):63–76, 2007.

[12] T. Berthold. *Primal heuristics for mixed integer programs*. PhD thesis, Zuse Institute Berlin (ZIB), September 2006.

[13] T. Berthold. Rens : The optimal rounding. *Mathematical Programming Computation*, 6(1):33–54, 2013.

[14] T. Berthold. *Heuristic algorithms in global MINLP solvers*. Verlag Dr. Hut, 2015.

[15] M. O. Bibi. *Methods for solving linear-quadratic problems of optimal control*. PhD thesis, University of Minsk, 1985.

[16] M. O. Bibi. *Cours de Master 1 en programmation linéaire et quadratique*. Université de Béjaia, 2015.

[17] M. O. Bibi and M. Bentobache. A hybrid direction algorithm for solving linear programs. *International Journal of Computer Mathematics*, 92(1):201–216, 2015.

[18] M. O. Bibi, H. Boussouira, and A. Laouar. Solution of an integer linear programming problem via a primal dual method combined with a heuristic. *International Journal of Mathematical Modelling and Numerical Optimisation*, 12(1):69–87, 2022.

[19] A. Billionnet. *Optimisation Discrète, de la modélisation à la résolution par des logiciels de programmation mathématique*. January 2007.

[20] S. Boudjelda and B. Brahmi. Parametric support approach for solving mean-variance problem under general constraints. *Statistics, Optimization & Information Computing*, 13(1):189–208, 2025.

[21] H. Boussouira. *Méthode de Support pour la Résolution des Problèmes de Programmation Linéaire en Nombres Entiers et Mixtes*. Thèse de magistère, Université de Béjaia, Béjaia, Octobre 2009.

[22] H. Boussouira and M. O. Bibi. Solving mixed integer linear programming problems with bounded variables by adaptive method. *International Journal of Mathematics in Operational Research*, 2024.

[23] S. S. Brito and H. G. Santos. Preprocessing and cutting planes with conflict graphs. *Computers & Operations Research*, 128:105176, 2021.

[24] H. Crowder, E. L. Johnson, and M. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, 1983.

[25] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[26] G. B. Dantzig. Linear programming and extensions. *Princeton University Press, Princeton, N.J.*, 1963.

[27] G. B. Dantzig and M. N. Thapa. *Linear Programming 1 : Introduction*. Springer, November 2013.

[28] J. Y. Ding, C. Zhang, L. Shen, S. Li, B. Wang, Y. Xu, and L. Song. Accelerating primal solution findings for mixed integer programs based on solution prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1452–1459. Association for the Advancement of Artificial Intelligence (AAAI), 2020.

[29] K. Djeloud, M. Bentobache, and M. O. Bibi. A new method with hybrid direction for linear programming. *Concurrency and Computation : Practice and Experience*, 33(1):e5836, 2021.

[30] M. Fischetti and M. Monaci. Proximity search for 0-1 mixed-integer convex programming. *Journal of Heuristics*, 20(6):709–731, 2014.

[31] J. B. J. Fourier. Solution d'une question particuliere du calcul des inégalités, and extracts from histoire de lacademie (1823, 1824). *Oeuvres II*, pages 317–328, 1826.

[32] R. Gabasov and F. M. Kirillova. *Methods of Linear Programming*, volume 1, 2 and 3. Edition of the Minsk University, Minsk, 1977, 1978 and 1980 (in Russian).

[33] R. Gabasov, F. M. Kirillova, and S. V. Prischepova. *Optimal feedback control*. Number 207. Springer Berlin, Heidelberg, 1995.

[34] G. Gamrath, T. Koch, A. Martin, M. Miltenberger, and D. Weninger. Progress in presolving for mixed integer programming. *Mathematical Programming Computation*, 7(4):367–398, 2015.

[35] P. Gemander, W. K. Chen, D. Weninger, L. Gottwald, A. Gleixner, and A. Martin. Two-row and two-column mixed-integer presolve using hashing-based pairing methods. *EURO Journal on Computational Optimization*, 8(3–4):205–240, 2020.

[36] A. Gleixner, L. Gottwald, and A. Hoen. Papilo : A parallel presolving library for integer and linear optimization with multiprecision support. *INFORMS Journal on Computing*, 35(6):1329–1341, 2023.

[37] R. E. Gomory. An algorithm for the mixed integer problem. *Report No. P-1885, The Rand Corporation, Santa Monica, CA.*, 1960.

[38] N. Gould and P. L. Toint. Preprocessing for quadratic programming. *Mathematical Programming*, 100(1):95–132, 2004.

[39] R. Guerbane. *Méthode primale-duale de programmation linéaire avec direction hybride*. Thèse de doctorat, Université de Béjaia, Béjaia, March 2023.

[40] R. Guerbane and M. O. Bibi. Primal-dual method for a linear program with hybrid direction. *International Journal of Mathematics in Operational Research*, 23(3):316–343, 2022.

[41] M. Guignard and K. Spielberg. Logical reduction methods in zero-one programmingminimal preferred variables. *Operations Research*, 29(1):49–74, February 1981.

[42] K. Halbig, A. Hoen, A. Gleixner, J. Witzig, and D. Weninger. A diving heuristic for mixed-integer problems with unbounded semi-continuous variables, 2024.

[43] G. Hendel. *New rounding and propagation heuristics for mixed integer programming*. PhD thesis, Zuse Institute Berlin (ZIB), Berlin, 2011.

[44] K. L. Hoffman and M. Padberg. Improving LP-representations of zero-one linear programs for branch-and-cut. *ORSA Journal on Computing*, 3(2):121–134, 1991.

[45] Z Hua, B Zhang, and X Xu. A new variable reduction technique for convex integer quadratic programs. *Applied Mathematical Modelling*, 32(2):224–231, 2008.

[46] E. L. Johnson and U. H. Suhl. Experiments in integer programming. *Discrete Applied Mathematics*, 2(1):39–55, 1980.

[47] L. V. Kantorovich. Mathematical methods of organizing and planning production. *Management science*, 6(4):366–422, 1960.

[48] F. M. Kirillova, R. Gabasov, and O. I. Kostyukova. A method of solving general linear programming problems. *Doklady AN BSSR (in Russian)*, 23:197–200, 1979.

[49] V. Klee and G. J. Minty. How good is the simplex algorithm ? *Inequalities III, Academic Press, New York*, pages 159–175, 1972.

[50] T. Kleinert, J. Manns, M. Schmidt, and D. Weninger. Presolving linear bilevel optimization problems. *EURO Journal on Computational Optimization*, 9:100020, 2021.

[51] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. Miplib 2010 : Mixed integer programming library version 5. *Mathematical Programming Computation*, 3(2):103–163, 2011.

[52] B. Kolman and R. E. Beck. *Elementary linear programming with applications*. Academic Press, 1995.

[53] A. Laouar. *Méthode adaptée pour la résolution d'un problème de programmation quadratique convexe à variables mixtes*. Thèse de magistère, Université de Béjaia, Béjaia, Janvier 2010.

[54] A. Laouar and M. O. Bibi. An adaptive hybrid direction algorithm for convex box-QP problems with enhanced pre-solving. *International Journal of Mathematics in Operational Research*, 2024.

[55] S. J. Leon, L. De Pillis, and L. G. De Pillis. *Linear algebra with applications*. Pearson Prentice Hall Upper Saddle River, NJ, 2006.

[56] F. Matter and M. E. Pfetsch. Presolving for mixed-integer semidefinite optimization. *INFORMS Journal on Optimization*, 5(2):131–154, 2023.

[57] T. Melo, S. Nickel, and F. Saldanha-da Gama. An LP-rounding heuristic to solve a multi-period facility relocation problem. Technical Report 168, Fraunhofer (ITWM), 2009.

[58] C. Mészáros and U. H. Suhl. Advanced preprocessing techniques for linear and quadratic programming. *OR Spectrum*, 25(4):575–595, 2003.

[59] A. Ndayikengurutse and S. Huang. Implementation of presolving and interior-point algorithm for linear & mixed integer programming : Software. *Journal of Systems Science and Information*, 8(3):195–223, 2020.

[60] A. Rezzag and M. O. Bibi. Procédure de presolving en programmation linéaire. In *Séminaire Mathématique de Béjaia*. Unité de recherche LaMOS (Modélisation et Optimisation des systèmes). Université de Béjaia, October 2022.

[61] A. Rezzag and M. O. Bibi. Procédure de presolving en programmation linéaire en nombres binaires. In *Colloque International Méthodes et Outils d'Aide à la Décision MOAD'2022*. Université de Béjaia, November 2022.

[62] A. Rezzag and M. O. Bibi. Reduction of the bounds for ILP problems. In *The 4th National Conference of Mathematics and Applications CNMA - 2024*. University center of Mila, December 2024.

[63] A. Rezzag and M. O. Bibi. Variable reduction for ILP problems : Method and analysis with gurobi. In *The first National Conference on Applied Mathematics, Statistics and Applications NCAMSA'24*. University of Batna 2, November 2024.

[64] A. Rezzag, M. O. Bibi, and A. Laouar. Reduction of bounded variables in integer linear programming problems. *International Journal of Mathematics in Operational Research*, 30(2):251–265, 2025.

[65] A. Rezzag, M. O. Bibi, and H. Ouzia. Rounding heuristic for integer linear programs. In *The 17th African Conference on Research in Computer Science and Applied Mathematics - Digital Sciences in Africa CARI'2024*. University of Bejaia, November 2024.

[66] Y. Shen, Y. Sun, A. Eberhard, and X. Li. Learning primal heuristics for mixed integer programs. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[67] H. P. Williams. A reduction procedure for linear and integer programming models. In *Redundancy in Mathematical Programming*, pages 87–107. Springer Berlin Heidelberg, 1983.

[68] H. P. Williams. The elimination of integer variables. *Journal of the Operational Research Society*, 43(5):387–393, 1992.

[69] J. Witzig and A. Gleixner. Conflict-driven heuristics for mixed integer programming. *INFORMS Journal on Computing*, 33(2):706–720, 2021.

[70] H. Zhang and J. Yuan. A combined variable aggregation presolving technique for mixed integer programming. *Operations Research Letters*, 53:107074, 2024.

[71] M. Zhou and W. Chen. Reducing the number of variables in integer quadratic programming problem. *Applied mathematical modelling*, 34(2):424–436, 2010.

[72] N. Zhu and K. Broughan. A note on reducing the number of variables in integer programming problems. *Computational Optimization and Applications*, 8(3):263–272, 1997.

[73] N. Zhu and K. Broughan. On dominated terms in the general knapsack problem. *Operations Research Letters*, 21(1):31–37, 1997.

RÉSUMÉ

Dans cette thèse, nous nous sommes intéressés à l'amélioration des méthodes de résolution pour les problèmes de Programmation Linéaire en Nombres Entiers (PLNE) à variables bornées, en combinant des techniques de presolving et une heuristique basée sur une procédure d'arrondissement. Le presolving vise à réduire la taille des modèles en fixant certaines variables, en éliminant les redondances, en resserrant les bornes des variables et en simplifiant les contraintes, tandis que l'heuristique propose une méthode rapide pour obtenir des solutions entières réalisables, et ce, en résolvant un problème auxiliaire.

Les résultats expérimentaux montrent que le presolving simplifie efficacement les modèles et améliore les performances des solveurs, tandis que l'heuristique offre un compromis intéressant entre rapidité d'exécution et qualité des solutions, avec des écarts satisfaisants par rapport à l'optimalité. Ces contributions constituent une avancée pour résoudre efficacement les PLNE de grande taille, tout en ouvrant des perspectives pour des améliorations futures.

Mots clés : Programmation linéaire en nombres entiers ; Variales bornées ; Relaxation ; Procédure de presolving ; Méthode heuristique ; Arrondissement ; Gurobi

---

ABSTRACT

In this thesis, we focused on improving resolution methods for bounded Integer Linear Programming (ILP) problems by combining presolving techniques and a heuristic based on a rounding procedure. Presolving aims to reduce the size of models by fixing certain variables, eliminating redundancies, tightening variable bounds, and simplifying constraints, while the heuristic provides a fast approach to obtain feasible integer solutions and that by solving an auxiliary problem.

Experimental results show that presolving effectively simplifies models and enhances solvers performances, while the heuristic offers an interesting trade-off between execution speed and quality of the solutions, with satisfactory gaps according to the optimality. These contributions represent a significant step forward in efficiently solving large-scale ILPs, while also paving the way for future improvements.

Key words : Integer linear programming ; Bounded variables ; Relaxation ; Presolving procedure ; Heuristic method ; Rounding ; Gurobi

---

ملخص

في هذه الأطروحة, ركزنا على تحسين طرق حل البرمجة الخطية في الأعداد الصحيحة ذات المتغيرات المحدودة من خلال الجمع بين تقنيات التمهيد و طريقة استدلالية تعتمد على التقريب. يهدف التمهيد إلى تقليل حجم النماذج من خلال تثبيت بعض المتغيرات, وإزالة التكرار, وتصغير حدود المتغيرات, وتبسيط القيود, بينما توفر الخوارزمية نهجا سريعا للحصول على حلول صحيحة و مقبولة باستخدام نموذج مساعد.

تظهر النتائج التجريبية أن التمهيد يبسط النماذج بفعالية و يعزز أداء المحلل, بينما تقدم الخوارزمية توازنا بين سرعة التنفيذ وجودة الحلول, مع تحقيق فجوات مرضية مقارنة بالقيم المثلى. تمثل هذه المساهمات خطوة مهمة نحو حل مشاكل البرمجة الخطية في الأعداد الصحيحة ذات أحجام كبيرة بكفاءة، مع فتح المجال لتحسينات مستقبلية.

كلمات المفتاح : البرمجة الخطية في الأعداد الصحيحة ; المتغيرات المحدودة ; التمهيد ; طريقة استدلالية ; التقريب ; قوروبي

AGZUL

Deg ugemmir ayi, nerra-d tamuγli nneγ ar usnefli n tarrayin n tifrat i yegna n usmihel imzireg es yemḍanen ummiden aked imuttiyen ugmiren. I waya, neddes titiktikin n tezwart aked tarrayt tagacurant ireṣṣan γef ubrid n tayaẓt. Tifrat n tezwart teţţawi γer usemẓi n tiddi n yemsilen, i d-yellan es ureṣṣi n kra imuttiyen, es tukksa n wallus, s usemẓi n yegmiren n imuttiyen aked usifses n tmariwin, skud tarrayt tagacurant teţţak-ed tifratin tummidin yeţţwaqbalen, ayagi mi'ara nefru yiwen wegnu amalel.

Igmuḍen umḍinen sseknen-d belli tifrat n tezwart tessifsus imsilen es tmellilt u teţţseggim tizemmar n imefruyen, skud tarrayt tagacurant teţţak-ed amnekni yelhan ger tizerzert uselkem aked tγara n tefratin, es wannufen yessillizen es wassaγ γer takkayit. Igmuḍen ayi gan d asurif axatar di tifrat n yegna n PLNE at tiddi tameqqrant, es tullya n webrid i-usnerni γer zdat.

Awalen n tsura : Asmihel imzireg es yemḍanen ummiden ; Imuttiyen ugmiren ; Alway ; Tifrat n tezwart ; Tarrayt tagacurant ; Asiẓi ; Gurobi