République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ YAHIA FARÈS DE MÉDÉA

Faculté des Sciences

Département de Mathématiques et l'informatique

MÉMOIRE DE MASTER

en mathématiques

Option : Recherche Opérationnelle

Problème de localisation d'entrepôts avec capacité

MODÉLISATION ET TECHNIQUES DE RÉSOLUTION

Présenté

par

BENTAIEB Khaoula et ZOUAOUAT Wiam

Soutenu publiquement devant le Jury composé de :

N. IKHLEF ESCHOUF	Professeur	UYF – Médéa	Président
S. KOUIDER AÏAD	MAA	UYF – Médéa	Examinatrice
A. BENKACI	MAA	UYF – Médéa	Promoteur

Médéa, 17 juin 2023

Résumé

Dans ce mémoire, nous nous intéressons au problème de localisation d'entrepôts avec capacité (CFLP). Dans un tel problème, un certain nombre entrepôts capacitifs doivent être localisés parmi *m* sites potentiels afin de satisfaire les demandes de *n* clients, en minimisant le coût total d'affectation (client - entrepôt) ainsi que les charges fixes d'établissement de ces entrepôts. Ce problème est issu du domaine de gestion et logistique. Il est formulé comme un problème de programmation linéaire en nombres binaires (0/1). Pour le résoudre, nous adoptons une approche polyèdrale en appliquant la technique des coupes planes (CP)[14] en conjonction avec un algorithme de Séparation et Évaluation Progressive (SEP) [15]. Un programme informatique a été élaboré, en utilisant la composante logicielle SOLVEUR implémentée sous l'application EXCEL. Nous reportons des résultats numériques obtenus en testant ce programme sur quelques exemples du problème. Ces résultats confirment l'efficacité de cette approche.

Mots–clés : localisation d'entrepôts, programmation en nombre entiers, méthode de coupes planes.

Abstract

In this report, we are interested in the capacitated facility location problem (CFLP). In such a problem, a number of capacitated facilities must be located among *m* potential sites in order to satisfy the demands of *n* customers, while minimizing the total allocation cost (customer - facility) as well as the fixed costs of establishing these facilities. This problem comes from the field of management and logistics. It is formulated as a linear programming problem in binary numbers (0/1). To solve it, we adopt a polyhedral approach by applying the technique of plane cuts (PC)[14] in conjunction with a Branch and Bound algorithm (B&B)[15]. A computer program has been developed, using the SOLVER software component implemented under the EXCEL application. We report numerical results obtained by testing this program on some examples of the problem. These results confirm the efficiency of this approach.

Keywords : facilities location, integer programming, cutting plane method.

Remerciements

Nous remercions ALLAH le Tout Puissant, pour nous avoir donné le courage et la force pour réaliser ce travail de recherche.

Nous tenons à remercier notre promoteur Monsieur A. BENKACI, enseignant à la faculté des sciences de l'université de Médéa, qui nous a encadrées tout au long de ce mémoire et qui nous a fait partager ses idées et ses intuitions. Qu'il soit aussi remercié pour sa disponibilité permanente et pour les nombreux encouragements qu'il nous a dépensés.

Nos vifs remerciements vont à notre enseignante, Madame S. KOUIDER AÏAD, enseignante à l'université de Médéa, d'avoir acceptée d'examiner notre travail, et aussi pour nous avoir fait partager ses connaissances durant toute la formation de Master Recherche Opérationnelle.

Nous sommes très honorées par la présence au jury notre enseignant, Monsieur N. IKHLEF-ESCHOUF, Professeur à l'université de Médéa. Nous le remercions très sincèrement, d'avoir accepté d'être président du jury.

Enfin, nous remercions toutes les personnes qui ont contribuées au succès de notre formation et qui nous ont aidées lors de la rédaction de ce mémoire.

Médéa, le 17 juin 2023

Table des matières

Page

RÉSUMÉ i

ABSTRACT ii

REMERCIEMENTS iii

LIST OF FIGURES vi

LIST OF TABLES vii

INTRODUCTION 9

CHAPITRE

1 NOTIONS PRÉLIMINAIRES SUR L'OPTIMISATION COMBINATOIRE 12

1.1 NOTIONS PRÉLIMINAIRES SUR LES GRAPHES 12

1.1.1 VOCABULAIRE 12

1.1.2 GRAPHE PARTIEL ET SOUS-GRAPHE 14

1.1.3 MATRICE D'ADJACENCE ASSOCIÉE À UN GRAPHE . . . 14

1.1.4 DIFFÉRENTS TYPES DE GRAPHES 16

1.2 COLORATION DE GRAPHES 17

1.3 PROGRAMMATION LINÉAIRE EN NOMBRES ENTIERS 18

1.3.1 QUALITÉ DE FORMULATION ALGÉBRIQUE (LARGE OU SERRÉE) 20

1.3.2 INÉGALITÉ VALIDE ET COUPE VALIDE 20

1.3.3 EXEMPLE (PROBLÈME DU STABLE DE TAILLE MAXIMUM) 21

2 MÉTHODES DE RÉSOLUTION ET COMPLEXITÉ 23

2.1 COMPLEXITÉ DE PROBLÈMES COMBINATOIRES 23

2.2 MÉTHODES DE RÉSOLUTION 25

2.2.1 Méthodes exactes 25
2.2.2 Méthodes approchées 26

3 Localisation d'entrepôt avec contrainte de capacité 29
3.1 Description du Problème (CFLP) 30
3.1.1 Paramètres et notations 30
3.1.2 Hypothèses 31
3.1.3 Critère d'optimisation 32
3.1.4 Motivation et Importance pratique 32
3.2 Modèle mathématique 33
3.2.1 Dérivation d'une coupe plane 34
3.2.2 Validations des données 35
3.3 Processus de résolution 36

4 Implémentation et tests numériques 37
4.1 Exemple Test 37
4.1.1 Matrice de distances (entrepôt-client) 37
4.1.2 Demandes de clients 37
4.1.3 Couts fixes et capacités des entrepôts 38
4.2 Approche de résolution 38
4.2.1 Relaxation totale des contraintes d'intégrité 38
4.2.2 Relaxation partielle des contraintes d'intégrité 38
4.2.3 Coupe plane valide 40
4.2.4 Algorithmes 40
4.3 Résultats numériques et interprétation 41

Conclusion 43

Notations et abréviations 45

Références 48

Liste des figures

1.1 Un graphe $G = (V, E)$ d'ordre 6 comportant 7 arêtes. 14

1.2 Graphe partiel de G. 14

1.3 Sous-graphe de G. 15

1.4 Graphe complet K_4. 16

1.5 Graphe biparti complet $K_{2,2}$ ou C_4. 16

1.6 Graphe biparti P_4. 17

2.1 Classification des problèmes combinatoires. 25

2.2 Schéma récapitulatif des méthodes de résolution des problèmes d'optimisation combinatoire. 28

3.1 Schématisation d'une solution du problème (CFLP). 34

3.2 Schéma conceptuel de résolution d'un problème réel. 36

4.1 Arborescence sans coupe valide 41

4.2 Arborescence avec coupe valide 42

Liste des tables

4.1 Coût d'affectation (entrepôt-client) 39

4.2 Demande de chaque client 39

4.3 Coût fixe et capacité de chaque entrepôts. 39

Liste des algorithmes

4.1 Relaxation partielle sans coupe valide. 40

4.2 Relaxation partielle avec coupe valide. 40

Introduction

L'optimisation combinatoire appelée aussi optimisation discrète, est une branche de mathématiques appliquées également liée à l'algorithmique et la théorie de la complexité. Elle occupe une place très importante dans la vie quotidienne. Son importance se justifie d'une part, par la grande difficulté des problèmes d'optimisation sous-jacents, et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire afin de modéliser de nombreux problèmes concrets dans différents secteurs de l'industrie telle que la logistique, la gestion,...,etc.

Un problème d'optimisation combinatoire consiste à trouver la meilleure solution selon un critère donné, dans un ensemble discret fini de grande taille, dit ensemble des solutions réalisables. Beaucoup de problème d'optimisation combinatoire dans la pratique, se modélise par un programme linéaire en nombres entiers.

Dans ce mémoire, nous nous intéressons au problème de localisation d'entrepôts avec capacité (CFLP). Dans un tel problème, un certain nombre entrepôts capacitifs doivent être localisés parmi m sites potentiels afin de satisfaire les demandes de n clients, en minimisant le coût total d'affectation (client - entrepôt) ainsi que les charges fixes d'établissement de ces entrepôts. Ce problème provenant du domaine de gestion et logistique, est formulé comme un problème de programmation linéaire en nombres binaires (0/1). Pour le résoudre, nous adoptons une approche polyèdrale en appliquant la méthode des coupes planes [14] en conjonction avec un algorithme de Séparation et Évaluation Progressive (SEP)[15].

La méthode de coupes planes (Cutting-Plane) est conçue pour résoudre des problèmes d'optimisation combinatoire qui se formulent sous la forme d'un programme linéaire en nombres entiers (PLNE). Cette méthode consiste à résoudre un problème relaxé, et à ajouter itérativement des contraintes au problème initial. Ces contraintes sont appelées des coupes planes. L'algorithme s'arrête lorsque la solution courante soit entière, on obtient ainsi une solution optimale pour le problème initial.

Un algorithme de séparation et évaluation (Branch and Bound) permet de déterminer par énumération implicite de l'ensemble de solutions réalisables, noté X d'un problème combinatoire, une solution qui optimise un certain critère donné. L'exploration de X du problème étudié repose sur deux principes :

- Un principe de séparation qui permet de réduire l'exploration de X à celle d'un sous ensemble de X de plus en plus réduit.
- Un principe d'évaluation qui consiste à définir une évaluation de la valeur des solutions, pour chaque sous ensemble de X obtenu par séparation.

Ce mémoire se décompose en une introduction générale, quatre chapitres et une conclusion générale.

Le premier chapitre aborde en premier temps quelques définitions sur les graphes simples. Ensuite les principaux concepts de la Programmation Linéaire en Nombres Entiers (PLNE) sont rappelés en mettant l'accent sur la notion de coupe valide, ainsi que leur développement mathématique qui va nous être utile tout au long de ce mémoire.

Dans le deuxième chapitre, nous rappelons en premier temps, les notions de base liées à la complexité de problèmes combinatoires, ensuite, nous exposons les méthodes de résolution exactes et approchées les plus utilisées dans la pratique, pour la résolution des problèmes combinatoire classés NP-difficile [9].

Dans le troisième chapitre, nous exposons d'abord, le problème de localisation d'entrepôts avec capacité. Nous évoquons aussi l'importance pratique du problème dans différents secteurs de la vie quotidienne. Ensuite, nous présentons un programme linéaire en

nombres bivalents (0/1), provenant de la littérature [7, 11, 14]. Sa résolution exacte est difficile et les tailles solvables sont assez réduites. Afin de valider les données du modèle mathématique adopté, des conditions nécessaires d'existence de solutions ont été établies. Nous dérivons aussi une coupe valide permettant de réduire l'espace de recherche pour ce modèle.

Le quatrième chapitre est consacré à l'exposition et l'implémentation de notre approche de résolution consistant à appliquer la méthode des coupes planes en conjonction avec un algorithme de Séparation et Évaluation Progressive. Nous reportons aussi des résultats numériques obtenus en testant notre programme sur un exemple type du problème (CFLP).

Au final, le manuscrit se termine par une conclusion générale, et des perspectives de recherche envisagées.

Chapitre 1

Notions préliminaires sur l'optimisation combinatoire

Dans ce chapitre, nous présentons tout d'abord quelques définitions sur les graphes simples, ensuite les principaux concepts de la Programmation Linéaire en Nombres Entiers (PLNE) sont rappelés en mettant l'accent sur la notion de coupe valide, ainsi que leur développement mathématique qui va nous être utile tout au long de ce mémoire.

1.1 Notions préliminaires sur les graphes

Dans cette section, nous présentons des définitions pour que le lecteur puisse se familiariser avec la terminologie utilllisée dans ce mémoire.

1.1.1 Vocabulaire

- Un graphe est défini par deux ensembles : un ensemble $V = \{v_1, v_2, \ldots, v_n\}$ dont les éléments sont appelés sommets, et un ensemble $E = \{e_1, e_2, \ldots, e_m\}$ dont les éléments sont appelés arêtes. On note $G = (V, E)$.
- Une arête e est une paire de sommets $(u, v) \in E \times E$. Les sommets u et v sont les extrémités de l'arête $e = (u, v)$.

- Deux sommets d'un graphe sont voisins ou adjacents s'ils ont une arête incidente en commun.
- Deux arêtes d'un graphe sont adjacentes si elles ont un sommet en commun.
- On appelle ordre d'un graphe le nombre de sommets de ce graphe.
- Le nombre d'arêtes incidentes à un sommet est le nombre d'arêtes sortant ou rentrant du sommet.
- Le degré d'un sommet v de G est le nombre d'arêtes incidentes à v. Il est noté $d(v)$. On note aussi :
- Le degré maximum $\Delta(G) = \max\{d(v) : v \in V\}$.
- Le degré minimum $\delta(G) = \min\{d(v) : v \in V\}$.
- Lorsque $d(v) = 0$, on dit que le sommet v est isolé. Lorsque $d(v) = 1$, on dit que le sommet v est pendant.
- L'ensemble de sommets voisins de v, est noté $\Gamma(v) = \{u \in V : (v, u) \in E\}$.

Exemple 1.1 Soit, le graphe $G = (V, E)$ représenté dans la Figure (1.1). Dans cet exemple :

- $V = \{1, 2, 3, 4, 5, 6\}$ et $E = \{(1, 2), (1, 6), (2, 3), (2, 6), (3, 4), (3, 5), (5, 6)\}$.
- Trois arêtes sont incidentes au sommet 2. Le sous-ensemble $\Gamma(2) = \{1, 3, 6\}$ représente les voisins du sommet 2, et $d(2) = |\Gamma(2)| = 3$.
- Le sommet 4 est pendant : $d(4) = 1$.
- Le degré maximum et minimum sont, respectivement : $\Delta(G) = 3$ et $\delta(G) = 1$.

Propriété 1.1 *On donne ici, deux propriétés très importantes,*

- *La somme de degrés des sommets d'un graphe est égale à deux fois son nombre d'arêtes* $\sum_{v \in V} d(v) = 2|E|$.
- *Le nombre de sommets de degré impair d'un graphe est pair.*

1.1.2 Graphe partiel et sous-graphe

Beaucoup de problèmes d'optimisation consistent à chercher un graphe partiel ou un sous-graphe vérifiant certaine propriété, dans un graphe G.

- Soit $G = (V, E)$ un graphe. Le graphe $G = (V, E')$ est un graphe partiel de G, si E' est inclus dans E.
- Un sous-graphe de G est un graphe $H = (V', E(V'))$ tel que V' est un sous ensemble de V, et $E(V')$ ont les arêtes induites par E sur V'.

La Figure (1.2), présente un graphe partiel et La Figure (1.3) represente un sous-graphe de G.

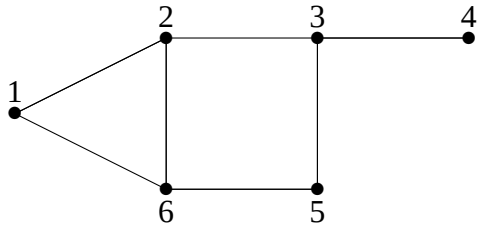


Figure 1.1 – Un graphe $G = (V, E)$ d'ordre 6 comportant 7 arêtes.

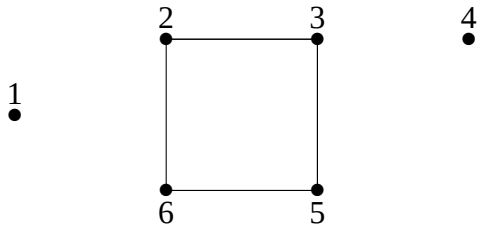


Figure 1.2 – Graphe partiel de G.

1.1.3 Matrice d'adjacence associée à un graphe

Parmi les matrices associées à un graphe donné, on cite la matrice d'adjacence. Elle joue un rôle déterminant pour la modélisation de problèmes d'optimisation.

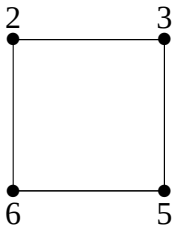


Figure 1.3 – Sous-graphe de G.

Définition 1.1 Soit $G = (V, E)$ un graphe dont les sommets sont numérotés de 1 à n. La matrice d'adjacence de G est la matrice carrée $\left[A_{ij}\right]_{(1\leq i,\, j\leq n)}$ définie par

$$A_{ij} := \begin{cases} 1 \text{ s'il y a une arête reliant le sommet } i \text{ au sommet } j \\ 0 \text{ sinon} \end{cases}$$

Propriété 1.2 *La matrice A est symétrique à diagonale nulle, en plus les égalités suivantes sont vérifiées :*

$$\sum_{j=1}^{n} A_{ij} = d\left(v_i\right), \quad i \in \{1, 2, \ldots, n\}, \tag{1.1}$$

$$\sum_{i=1}^{n} A_{ij} = d\left(v_j\right), \quad j \in \{1, 2, \ldots, n\}, \tag{1.2}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} = 2\left|E\right|. \tag{1.3}$$

La matrice d'adjacence associée à G, voir Figure (1.1), est la suivante :

$$A = \begin{pmatrix} 0 & 1 & & & & 1 \\ 1 & 0 & 1 & & & 1 \\ & 1 & 0 & 1 & 1 & \\ & & 1 & 0 & & \\ & & 1 & & 0 & 1 \\ 1 & 1 & & & 1 & 0 \end{pmatrix}$$

1.1.4 Différents types de graphes

Dans cette sous section, nous présentons quelques types de graphes, à savoir le graphe nul (ou vide), le graphe complet, le biparti et le biparti complet.

Graphe nul : Soit $G = (V, E)$ un graphe d'ordre n. Un sous-ensemble S de V est dit stable s'il ne comprend que des sommets non adjacents deux à deux. Il est noté I_n.

Graphe complet (ou clique) : Un graphe complet est un graphe où chaque sommet est relié à tous les autres. Le graphe complet d'ordre n est noté K_n. Dans ce graphe chaque sommet est de degré $n - 1$.

Graphe biparti : Un graphe $G = (V, E)$ est biparti ou bipartite si l'ensemble de ses sommets V peut être partitioné en deux ensembles disjoints V_1 et V_2, de sorte que les arêtes relient uniquement des éléments de V_1 à des éléments de V_2.

Graphe biparti complet : Un graphe bipartite complet $G = (V_1 \cup V_2, E)$, noté $K_{p,q}$, est un graphe bipartite où chaque sommet de V_1 est relié à tous les sommets de V_2 par une arête. Ici, .$|V_1| = p$ et $|V_2| = q$.

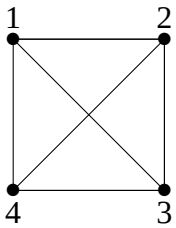


Figure 1.4 – Graphe complet K_4.

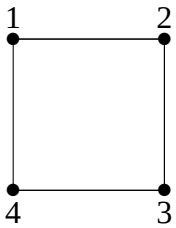


Figure 1.5 – Graphe biparti complet $K_{2,2}$ ou C_4.

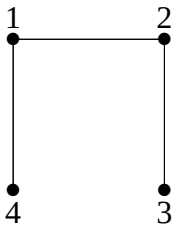


Figure 1.6 – Graphe biparti P_4.

1.2 Coloration de graphes

Dans la pratique, beaucoup de problèmes de gestion peuvent être modélisés comme un problème de coloration de graphes à savoir la répartition de services dans les réseaux, l'ordonnancement de tâches et l'allocation de fréquences,..., etc. Une *coloration propre* d'un graphe G est une fonction associant à tout sommet v dans V du graphe G un entier representant une couleur dans $\{1, \ldots, |V|\}$, telle que deux sommets adjacents n'aient pas la même couleur.

Le problème d'optimisation classique est de trouver une coloration avec un nombre de couleurs minimum. *Le nombre chromatique* correspond à ce nombre minimal.

Définition 1.2 (Nombre chromatique) Le nombre chromatique d'un graphe G, noté $\chi(G)$ est le nombre minimum de couleurs nécessaires pour lui donner une coloration propre.

Le problème de décision défini par « déterminer si $(\chi(G) \leq k)$? » est connu pour être NP-complet (voir chapitre 2). De plus une borne classique sur le nombre chromatique est la suivante.

Théorème 1.3 (Brooks, 1941) *Pour tout graphe G de degré maximum $\Delta(G)$, l'inégalité ci-dessous, est valide*

$$\chi(G) \leq \Delta(G) + 1.$$

À partir de ce résultat, de nombreux algorithmes ont été développés dans la littérature pour approximer $\chi(G)$.

La remarque suivante, est importante dans la résolution de problèmes d'optimisation d'ordre pratique.

Remarque 1.4 Une k-coloration est une partition de $V(G)$ en k ensembles indépendants (ou stables).

1.3 Programmation linéaire en nombres entiers

Dans cette section, nous rappelons brièvement les outils de base de la programmation en nombres entiers. On se focalise sur la notion de la coupe valide.

Tout programme linéaire en nombres entiers (PLNE) peut s'écrire ainsi :

$$(PLNE) \quad \begin{cases} \max cx \\ x \in X = P \cap \mathbb{Z}^n \end{cases}$$

où, $P = \{x \in \mathbb{R}^n : Ax \leq b \text{ et } x \geq 0\}$ est un polytope (polyèdre borné et fermé).

Définition 1.3 (Enveloppe convexe) L'enveloppe convexe de X, notée $\text{conv}(X)$, est le plus petit ensemble convexe contenant toutes solutions de X, et on note : $\text{conv}(X) = \left\{x \in \mathbb{Z}^n : A'x \leq b', x \geq 0\right\}$.

Un problème de programmation linéaire (PL) est un problème d'optimisation où la fonction économique et les contraintes sont toutes linéaires. Un problème de programmation linéaire en nombres entiers (PLNE) est un (PL) dans lequel les variables doivent

prendre des valeurs entières. Un (PLNE) s'écrit comme suit :

$$(PLNE) \quad \begin{cases} \max cx \\ Ax \leq b \\ x \in \mathbb{Z}_+^n \end{cases}$$

Avec $c \in \mathbb{Z}^n$, $b \in \mathbb{Z}^m$ et $A \in \mathbb{Z}^{mn}$.

Les éléments de l'ensemble $S = \{x \in \mathbb{Z}_+^n : Ax \leq b\}$ sont les solutions réalisables du problème,c'est-à-dire, les valeurs de x qui satisfont toutes les contraintes du modèle. conv(S) est l'enveloppe convexe de l'ensemble S. Les éléments de l'ensemble S qui maximisent la fonction objectif sont les solutions optimales.

Nous considérons les trois valeurs optimales des problèmes d'optimisation suivants :

1. Problème originaire : $Z_X = \max\{cx : x \in X\}$.
2. Problème équivalent : $Z_{\text{conv}(X)} = \max\{cx : x \in \text{conv}(X)\}$.
3. Problème relaxé : $Z_P = \max\{cx : x \in P\}$.

Propriété 1.5 *Nous pouvons montrer que,*

(i) *La valeur Z_p est une borne supérieure de Z_X : $Z_X \leq Z_P$.*

(ii) *Les points extrêmes de* conv(X) *sont des solutions de X : $Z_X =$* conv(X).

(iii) *Optimiser une fonction linéaire sur le domaine* conv(X) *est la formulation idéale pour le problème d'optimisation* $(PLNE)$.

Remarque 1.6 Les propriétés (ii) et (iii) sont très importantes. Malheureusement, la détermination de l'enveloppe convexe pour l'ensemble X est un problème aussi difficile que le problème d'optimisation (PLNE) originaire.

1.3.1 Qualité de formulation algébrique (Large ou Serrée)

Un problème d'optimisation peut avoir plusieurs formulations différentes mais équivalentes. On pratique, il est judicieux d'adopter la bonne formulation possible servant à faciliter la résolution du problème. On parle de la formulation serrée.

- Une bonne formulation est une formulation pour laquelle le polyèdre associé à la relaxation linéaire est proche de l'enveloppe convexe des solutions du problème, en particulier autour de la solution optimale.
- La relaxation linéaire d'une formulation serrée procure de meilleures bornes sur la solution optimale du problème.
- Une formulation serrée peut accélérer la convergence d'un algorithme de Séparation et Évaluation Progressive basé sur un programme linéaire relaxé.

1.3.2 Inégalité valide et coupe valide

Ici, on rappelle les définitions de base d'inégalité valide et de coupe valide. Le but est de déterminer une coupe valide associée un polyèdre P donné.

- Une inégalité : $\alpha x \leq \beta$ est dite valide pour le polytope P, si et seulement si, $P \cap \{x : \alpha x \leq \beta\} = P$.
- Une inégalité : $\alpha x \leq \beta$ est dite coupe pour le polytope P, si et seulement si, $P \cap \{x : \alpha x \leq \beta\} \subset P$ (inclusion stricte).
- Une inégalité : $\alpha x \leq \beta$ est dite coupe valide pour le polytope P si et seulement si, $\text{conv}(X) \subseteq P \cap \{x : \alpha x \leq \beta\} \subset P$.

Remarque 1.7 Dans la littérature, il existe plusieurs types de coupe valide, on cite, entre autre, les inégalités valides de Chvatal-Gomory, la coupe valide de recouvrement, ... etc. Pour le plus ample d'informations sur ce sujet, nous invitons le lecteur à consulter les ouvrages spécialisés dans la matière [1, 5, 15].

1.3.3 Exemple (Problème du stable de taille maximum)

Le problème du stable de taille maximum défini sur le graphe G peut se modéliser par le programme linéaire en nombres binaires suivant.

$$(S) \quad \begin{cases} (\max) z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\ \begin{pmatrix} 2 & 1 & & & & 1 \\ 1 & 3 & 1 & & & 1 \\ & 1 & 3 & 1 & 1 & \\ & & 1 & 1 & & \\ & & 1 & & 2 & 1 \\ 1 & 1 & & & 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \leq \begin{pmatrix} 2 \\ 3 \\ 3 \\ 1 \\ 2 \\ 3 \end{pmatrix} \\ x \in \{0; 1\}^6 \end{cases}$$

La solution optimale $\hat{x}$ de (S) est le vecteur représentatif d'un stable de taille maximum. La valeur optimale $z(\hat{x})$ est la taille du stable obtenu. Ce paramètre est bien connu en théorie des graphes sous l'appellation « nombre de stabilité, qui est noté $\alpha(G)$ ».

En résolvant le problème relaxé de (S), qui est noté (S_{RL}).

$$(S_{RL}) \quad \begin{cases} (\max) z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\ \begin{pmatrix} 2 & 1 & & & & 1 \\ 1 & 3 & 1 & & & 1 \\ & 1 & 3 & 1 & 1 & \\ & & 1 & 1 & & \\ & & 1 & & 2 & 1 \\ 1 & 1 & & & 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \leq \begin{pmatrix} 2 \\ 3 \\ 3 \\ 1 \\ 2 \\ 3 \end{pmatrix} \\ x \in [0, 1]^6 \end{cases}$$

on trouve :

$$\begin{aligned} \hat{x}_{RL} &= (\tfrac{3}{5}, \tfrac{4}{5}, 0, 1, 1, 0), \\ z(\hat{x}_{RL}) &= 3 + \tfrac{2}{5}. \end{aligned}$$

Inégalités valides : Le graphe G de la figure (1.1) comporte :

- Une clique maximale d'ordre 3, alors : $x_1 + x_2 + x_6 \leq 1$ est une inégalité valide. On peut écrire : $z(\hat{x}) \leq 4$.
- Un cycle impair d'ordre 5, alors : $x_1 + x_2 + x_3 + x_5 + x_6 \leq 2$ est une inégalité valide. On peut écrire : $z(\hat{x}) \leq 3$.

Ces deux inégalités, que nous venons de présenter, sont des coupes valides pour le domaine relaxé du problème (S).

Au final, si on ajoute la contrainte : $x_1 + x_2 + x_6 \leq 1$ à (S), la solution du problème relaxé devient : $\hat{x}_{RL'} = (0, 1, 0, 1, 1, 0)$ et $Z(\hat{x}_{RL'}) = 3$ qui est optimale.

Chapitre 2

Méthodes de résolution et complexité

Dans ce chapitre, nous rappelons tout d'abord, des notions de base liées à la complexité de problèmes de manière générale, ensuite nous exposons les méthodes de résolution exactes et approchées les plus connues pour la résolution des problèmes d'optimisation combinatoire classés NP-difficile.

2.1 Complexité de problèmes combinatoires

La théorie de la complexité (Cook, 1971) a pour but d'analyser les coûts de résolution, en termes de temps de calcul, des problèmes d'optimisation combinatoire. Elle permet d'établir une classification des problèmes en plusieurs niveaux de difficulté. Parmi les classes de problèmes combinatoires, on distingue les problèmes de décision et les problèmes d'optimisation.

- Un problème d'optimisation est un problème pour lequel on doit chercher une solution admissible optimisant au mieux une fonction économique.

- Un problème de décision est un énoncé auquel la réponse peut être uniquement OUI ou NON. Chaque problème d'optimisation possède un problème de décision correspondant.

Par la suite, nous avons besoin de ces deux définitions :

- Un algorithme est dit polynômial si le nombre d'opérations élémentaires, nécessaires pour atteindre le résultat, est majoré par un polynôme de la taille des données n, par exemple $O(n^{\alpha})$ avec α est une constante positive indépendante de n .
- Un algorithme est dit exponentiel si le nombre d'opérations élémentaires est minoré par une forme exponentielle de la taille des données n, par exemple $O(a^n)$ avec $(a > 1)$ est une constante indépendante de n.

La classe des problèmes NP (Non déterministe Polynomial) est la classedes problèmes de décision pouvant être résolus par un algorithme polynomial sur une machine de Turing non déterministe (Garey, 1979) [9]. C'est-à-dire, les problèmes de décision pour lesquels il existe un algorithme tel que si on lui fournit une solution dont la réponse est OUI, il est capable de la vérifier en temps polynomial.

Parmi la classe NP, on distingue deux grandes sous-classes : la classe des problèmes polynomiaux (la classe P) et la classe des problèmes NP-Complet (la classe NPC).

La classe P est la classe des problèmes de décision solvables par un algorithme polynomial déterministe, c'est-à-dire, les problèmes de décision pour lesquels il existe un algorithme polynomial permettant de répondre à la question par un OUI. C'est la classe des problèmes les plus « faciles » de NP.

La classe NPC est la classe des problèmes dont il n'existe aucun algorithme polynomial pour les résoudre, sous l'hypothèse (P $\neq$ NP). Cependant, si on peut résoudre un problème de NPC en un temps polynomial, on obtient par transformation des algorithmes polynomiaux pour tous les problèmes de NPC.

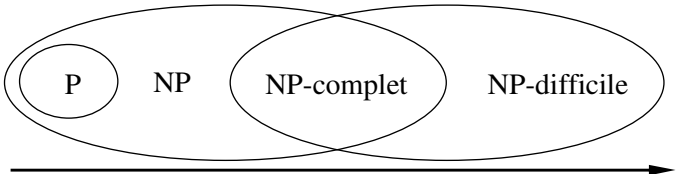


Figure 2.1 – Classification des problèmes combinatoires.

Quand un problème est NP-Complet, il n'est pas possible, à l'heure actuelle, de construire un algorithme polynomial pour le résoudre. Un problème d'optimisation est dit NP-Difficile si le problème de décision qui lui correspond est NP-Complet.

Pour résoudre un problème combinatoire de manière efficace, il faut prendre en compte la particularité de chaque problème. C'est pourquoi il y a autant d'approches que de problèmes. Toutefois, il existe des approches qui fournissent des solutions approchées de bonne qualité pour un certain nombre de problèmes.

2.2 Méthodes de résolution

Dans cette section, nous exposons les méthodes de résolution des problèmes combinatoires. Par la suite, nous présentons les méthodes de résolution les plus connues dans la littérature. Tout d'abord, nous commençons par les méthodes exactes, ensuite on présentera les méthodes approchées.

2.2.1 Méthodes exactes

En général, il n'est pas possible de construire un algorithme polynomial résolvant un problème d'optimisation NP-difficile. Toutefois, on peut mettre au point des méthodes efficaces pour une grande partie de problèmes. Pour certains, on appliquera une Procédure par Séparation et Évaluation Progressive (B&B), pour d'autres, la programmation

dynamique (DP) ou encore la programmation linéaire en nombres entiers (PLNE). Les méthodes exactes ne sont efficaces que pour les instances de problèmes de petite taille. Une référence dans le domaine est celle développée par [5, 8, 11, 13, 15, 16].

2.2.2 Méthodes approchées

Il existe deux types de méthodes approchées, à savoir les heuristiques et les méta-heuristiques.

2.2.2.1 Heuristique Le principe général d'une méthode heuristique est de trouver en un temps polynomial une solution réalisable la meilleure possible. Elle dépende de la nature du problème étudié.

2.2.2.2 Les méthodes constructives Le principe de ces méthodes est de construire une solution à partir des données initiales. Ce sont des méthodes itératives, où à chaque itération, on complète une solution partielle. Les méthodes constructives se distinguent par leur rapidité et leur grande simplicité. Le principal défaut de ces méthodes réside dans la qualité mauvaise des solutions obtenues. Le fait de vouloir opérer à tout prix le meilleur choix à chaque étape est une stratégie dont les effets peuvent être catastrophiques à long terme.

2.2.2.3 Méthodes amélioratrices Le principe de ces méthodes n'est plus de construire une solution à partir des données initiales du problème, mais de modifier le résultat d'une solution admissible en vue d'améliorer la valeur de la fonction objective. La plupart de ces méthodes utilisent la notion de voisinage de solution, on parle aussi de méthodes par voisinage.

À partir d'une solution initiale générée aléatoirement ou en appliquant une heuristique, une méthode par voisinage ou encore méthode de recherche locale réalise un processus

itératif qui consiste à remplacer la solution courante par l'un de ses voisins en tenant compte de la fonction objectif. La méthode s'arrête et retourne la meilleure solution trouvée. Le critère d'arrêt peut être un nombre d'itérations atteint ou une valeur de la fonction objective atteinte. Ces méthodes diffèrent essentiellement par le système de voisinage utilisé et la stratégie de parcours du système de voisinage.

2.2.2.4 Métaheuristique Une métaheuristique est un ensemble de concepts universels, applicables à une large classe de problèmes d'optimisation combinatoire. On distingue deux types : les méthodes à base d'individu, telles que le Recuit Simulé (Simulated Annealing SA) et la Recherche Tabou (Tabu Search, TS), et les méthodes à population telles que l'Algorithme Génétique (Genetic Algorithm, GA) et la Colonie de fourmis (Ant Colony Optimisation, ACO). Pour le plus amples d'informations, le lecteur peut consulter [6, 11, 15].

Nous présentons ci-dessus, un schéma général récapitulant les méthodes de résolution d'un problème d'optimisation combinatoire.

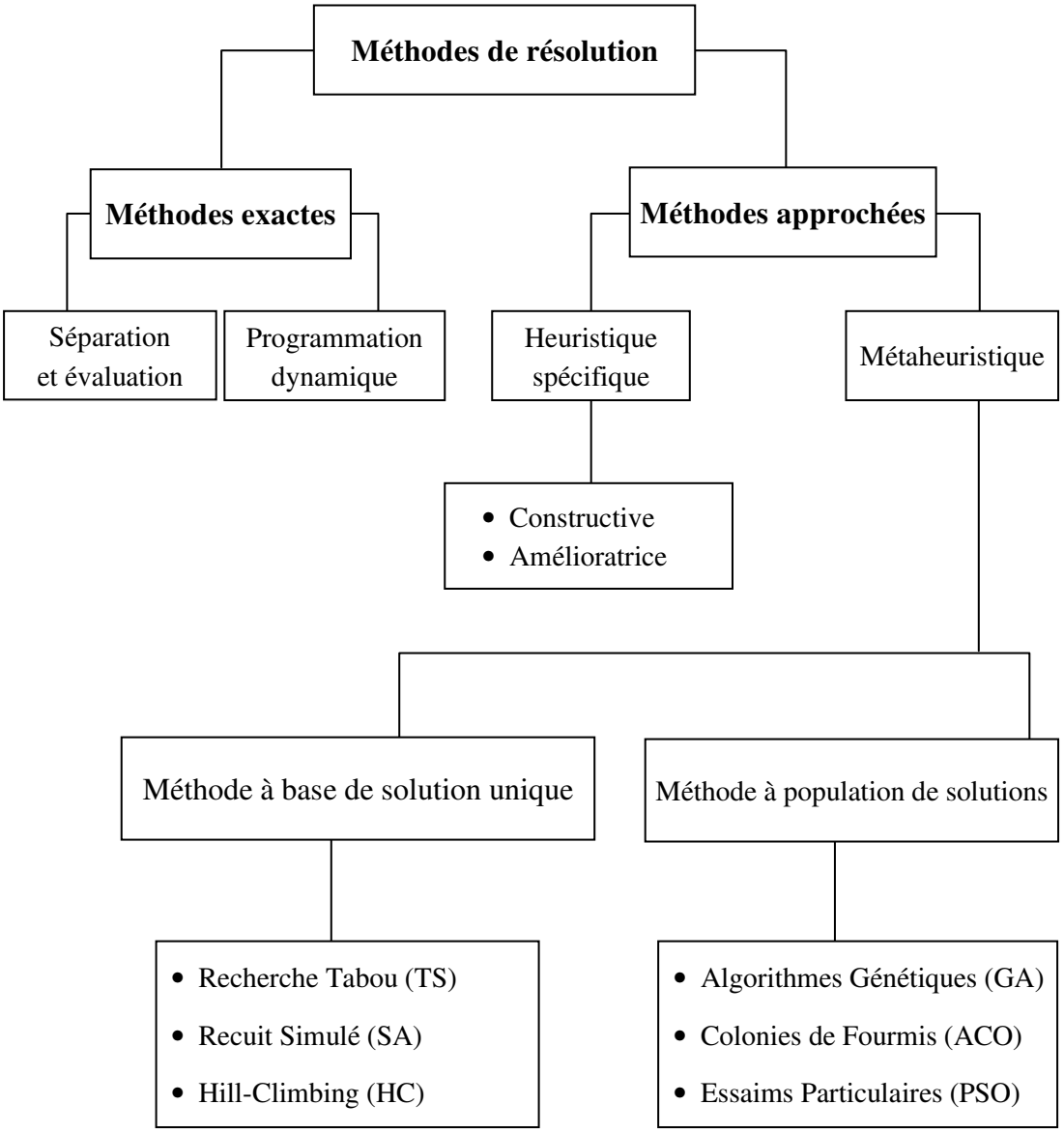


Figure 2.2 – Schéma récapitulatif des méthodes de résolution des problèmes d'optimisation combinatoire.

Chapitre 3

Localisation d'entrepôt avec contrainte de capacité

Le problème de localisation d'entrepôts avec capacité (CFLP) est un problème qui a été très étudié dans le domaine de la recherche opérationnelle depuis une dizaine d'années pour ses caractéristiques théoriques et ses applications dans le monde industriel. Il consiste à installer un certain nombre d'entrepôts sur un ensemble de sites potentiels et à affecter un et un seul entrepôt à chaque client d'un ensemble de clients. Sur chaque site, il coûte un certain montant pour y installer un entrepôt et pour chaque client, il coûte un autre montant pour l'affecter à un site sur lequel un entrepôt est installé. L'objectif est de minimiser le coût total. Ce problème classé NP-difficile est très compliqué à résoudre de manière exacte pour des instances de grande taille. Il a été largement étudié et résolu par un grand nombre de chercheurs dans le cas général par des heuristiques constructives et amélioratrices, ainsi que par des méta-heuristiques qui vont des méthodes à solution unique jusqu'aux méthodes à population de solutions.

Dans ce chapitre, nous exposons d'abord, le problème (CFLP). Nous évoquons aussi l'importance pratique du problème dans différents secteurs de la vie quotidienne. Ensuite, nous présentons un programme linéaire en nombres bivalents (0/1), provenant de la littérature. Sa résolution exacte est difficile et les tailles solvables sont assez réduites. Afin

de valider les données du modèle mathématique adopté, des conditions nécessaires d'existence de solutions ont été établies. Nous dérivons aussi une coupe valide permettant de réduire l'espace de recherche pour ce modèle.

3.1 Description du Problème (CFLP)

Cette section est consacrée à la position du problème de localisation d'entrepôts avec capacité. Il s'énonce comme suit.

Une entreprise fabriquant un produit cherche à s'implanter sur une zone géographique. L'entreprise a démarché n clients et prévoit sur un horison temporel à venir, de vendre d_j tonnes de produit à chaque client $j \in \{1, \ldots, n\}$. Elle dispose de m sites potentiels $S_1, S_2, \ldots, S_m$ pour installer ses entrepôts. Le coût d'installation d'un entrepôt sur le site S_k est évalué à F_k €, pour tout $k \in \{1, \ldots, m\}$. Un site S_k a une capacité de Q_k tonnes. Le coût d'affectation (client j - site S_k) est C_{kj} € par tonne. Le but est de déterminer les sites sur lesquels l'entreprise va établir ses entrepôts pour pouvoir satisfaire la demande de ses clients tout en minimisant le coût total.

3.1.1 Paramètres et notations

Nous utilisons les notations qui sont présentées dans la plupart des ouvrages et articles sur le sujet. Dans ce qui suit, nous allons adopter les notations suivantes :

- Sélectionner un ensemble d'entrepôts à localiser sur m sites potentiels pour desservir n clients.
- $D = \{1, \ldots, m\}$ désigne l'ensemble de sites potentiels.
- $C = \{1, \ldots, n\}$ désigne l'ensemble de clients.
- Il existe un coût fixe F_k pour la localisation de l'entrepôt k et celui-ci a une capacité maximale Q_k.

- d_j la demande du client j.
- C_{kj} le coût d'affectation de l'entrepôt k au client j.
- Variables de décision :

$$y_k = \begin{cases} 1, & \text{si l'entrepôt } k \text{ est choisie pour l'instalation.} \\ 0, & \text{sinon.} \end{cases} \tag{3.1}$$

et

$$x_{kj} = \begin{cases} 1, & \text{si le client } j \text{ est affecté à l'entrepôt } k. \\ 0, & \text{sinon.} \end{cases} \tag{3.2}$$

3.1.2 Hypothèses

Le problème classique de localisation d'entrepôts sans capacité (UFLP) a été principalement étudié avec des données déterministes et avec l'hypothèse que la capacité des installations peut répondre à la demande de tous les clients. Le problème (CFLP) comprend des hypothèses plus réalistes à savoir :

1. Toutes les données sont connues et déterministes, telles que les couts fixe F_k, les couts d'affectation C_{kj}, et les capacités Q_i d'entrepôts.
2. La demande d_j du client j est un paramètre àléatoire en pratique. Dans notre étude nous l'estimons par la formule ci-dessous,

$$d_j = \frac{d_j^{\min} + 4d_j^{\text{med}} + d_j^{\max}}{6} \tag{3.3}$$

   où, $d_j^{\min}$, d_j^{med} et $d_j^{\max}$ sont respectivement, la valeur minimale, médiane et maximale du paramètre d_j.
3. Un client peut être affecté à n'importe quel entrepôt. Pas de conflits entre les affectations de clients.

3.1.3 Critère d'optimisation

Le critère d'optimisation du problème (CFLP) se décompose en deux quantités à savoir le coût d'affectation (client-entrepôt) et le coût fixe d'instalation d'entrepôts.

- Coût d'affectation (client-entrepôt) :

$$\sum_{k=1}^{m}\sum_{j=1}^{n} C_{kj}x_{kj} \tag{3.4}$$

- Coût fixe d'instalation d'entrepôts :

$$\sum_{k=1}^{m} F_k y_k \tag{3.5}$$

Le critère d'optimisation de (CFLP) agrège les deux critères ci-dessus, et le coût total s'écrit ainsi,

$$Z(x,y) = \sum_{k=1}^{m} F_k y_k + \sum_{k=1}^{m}\sum_{j=1}^{n} C_{kj}x_{kj} \tag{3.6}$$

3.1.4 Motivation et Importance pratique

Les problèmes de localisation sont des problèmes de décisions qui font l'objet d'une littérature abondante et de nombreuses applications dans le monde réel. Ces problèmes traitent de la localisation d'un certain nombre d'installations pour approvisionner un ensemble de clients au coût minimum tout en respectant plusieurs contraintes et impliquent divers objectifs conflictuels.

Le problème de localisation d'installations avec capacité (CFLP) est l'un des modèles les plus importants et les plus fondamentaux, car il comprend des hypothèses plus réalistes que d'autres problèmes de localisation classiques. Le (CFLP) est classé NP-difficile, et il profondément étudié tant du point de vue de la modélisation mathématique que des algorithmes de résolution.

3.2 Modèle mathématique

Dans cette section, nous exposons un programme linéaire en nombres binaires (PLNB) qui est bien connu dans la littérature.

Le problème peut se formuler ainsi :

$$
\begin{aligned}
\min Z(x,y) = \quad & \sum_{k=1}^{m} F_k y_k + \sum_{k=1}^{m}\sum_{j=1}^{n} C_{kj} x_{kj} && && (a)\\
& \sum_{k=1}^{m} x_{kj} = 1, && j \in C && (b)\\
& \sum_{j=1}^{n} d_j x_{kj} \leq Q_k y_k, && k \in D && (c)\\
& x_{kj} \in \{0,1\}, && k \in D,\ j \in C && (d)\\
& y_k \in \{0,1\}. && k \in D && (e)
\end{aligned}
\tag{3.7}
$$

Dans le modèle (3.7),

(a) désigne le coût total qui est égal au coût fixe plus le coût d'affectation (client-entrepôt).

(b) chaque client j doit être affecté à un et un seul entrepôt.

(c) la demande de l'ensemble de clients affectés à l'entrepôt k, ne dépasse pas la capacité de ce dernier.

(d) si le client j est affecté à l'entrepôt k, $x_{kj} = 1$ sinon 0.

(e) si l'entrepôt k est choisie pour l'instalation, $y_k = 1$ sinon 0.

Il s'agit d'un programme linéaire en nombres binaires (PLNB). La taille de l'espace de recherche engendré par les variables de décision, est de l'ordre $2^{m(n+1)}$.

La résolution exacte de ce programme est difficile et les tailles des exemples du problème qui sont solvables exactement sont assez réduites.

Une solution du programme (3.7) peut etre schématisée par la Firgure (3.1).

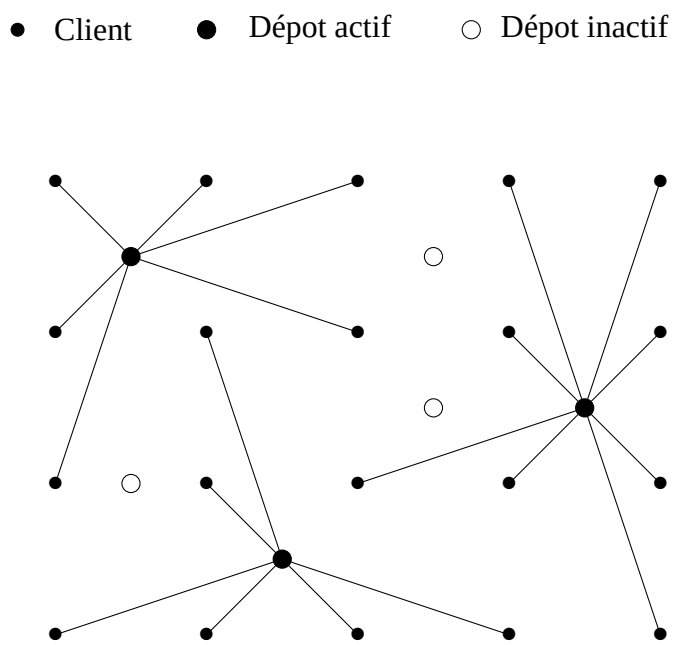


Figure 3.1 – Schématisation d'une solution du problème (CFLP).

3.2.1 Dérivation d'une coupe plane

Dans cette sous section, nous présentons la dérivation d'une coupe valide pour le modèle mathématique (3.7). Cette dernière permet de réduire l'espace de solutions du programme relaxé, ce qui implique une amélioration de la borne inférieure du programme (PLNE).

Proposition 3.1 *L'inégalité ci-dessous, constitue une coupe valide pour le programme mathématique* (3.7)

$$\sum_{k=1}^{m} y_k \geq \left\lceil \frac{1}{Q_{\max}} \sum_{j=1}^{n} d_j \right\rceil \tag{3.8}$$

Preuve. Nous agrègeons le groupe de contraintes (c) du programme (3.7), nous obtenons $\sum_{k=1}^{m}\sum_{j=1}^{n} d_j x_{kj} \leq \sum_{k=1}^{m} Q_k y_k$, puisque $\sum_{k=1}^{m} Q_k y_k \leq Q_{\max}\sum_{k=1}^{m} y_k$, l'inégalité précédente peut s'écrire $\sum_{j=1}^{n} d_j \sum_{k=1}^{m} x_{kj} \leq Q_{\max}\sum_{k=1}^{m} y_k$, et le fait que $\sum_{k=1}^{m} x_{kj} =$

1, $(\forall j \in C)$, nous obtenons l'inégalité $\sum_{j=1}^{n} d_j \leq Q_{\max} \sum_{k=1}^{m} y_k$, qui peut s'écrire ainsi $\sum_{k=1}^{m} y_k \geq \dfrac{1}{Q_{\max}} \sum_{j=1}^{n} d_j$. □

Remarque 3.2 Dans la pratique, la méthode des coupes planes converge lentement. Dans le chapitre 4, nous allons voir comment nous pouvons la combiner avec la méthode de Séparation et Évaluation (SEP).

3.2.2 Validations des données

Ici, nous présentons des conditions nécessaires d'éxistence de solutions du modèle mathématique (3.7).

Proposition 3.3 *Pour que le programme mathématique* (3.7) *ait une solution, il faut que les deux conditions ci-dessous, soient vérifiées.*

(α) $Q_{\max} \geq \frac{1}{m} \sum_{j \in C} d_j$,

(β) $d_{\min} \leq \frac{1}{n} \sum_{k \in D} Q_k$.

Preuve.

Pour (α), nous appliquons l'inégalité de la coupe valide (3.8), nous obtenons $\sum_{k=1}^{m} y_k \geq \frac{1}{Q_{\max}} \sum_{j=1}^{n} d_j$, et le fait que $\sum_{k=1}^{m} y_k \leq m$, l'inégalité peut s'écrire $m \geq \frac{1}{Q_{\max}} \sum_{j=1}^{n} d_j$, ou encore $Q_{\max} \geq \frac{1}{m} \sum_{j=1}^{n} d_j$.

Pour (β), nous agrègeons le groupe de contraintes (c) du programme (3.7), nous obtenons $\sum_{k=1}^{m} \sum_{j=1}^{n} d_j x_{kj} \leq \sum_{k=1}^{m} Q_k y_k$, puisque $\sum_{k=1}^{m} Q_k y_k \leq \sum_{k=1}^{m} Q_k$, l'inégalité précédente peut s'écrire $\sum_{j=1}^{n} d_j \sum_{k=1}^{m} x_{kj} \leq \sum_{k=1}^{m} Q_k$, et le fait que $\sum_{k=1}^{m} x_{kj} = 1$, $(\forall j \in C)$, nous obtenons $n d_{\min} \leq \sum_{j=1}^{n} d_j \leq \sum_{k=1}^{m} Q_k$, ce qui achève la preuve. □

Ces conditions jouent un rôle important dans la vérification de la solution, car elle permettent au gestionnaire, sans avoir recourir à la résolution, d'éliminer les jeux de données pour lesquels le programme (3.7) n'admette pas de solutions.

3.3 Processus de résolution

La résolution de notre problème passe par quatre étapes importantes qui se succèdent. Ces dernières peuvent être schématisées par le diagramme représenté dans la Figure (3.2).

Étape 1 Modélisation : Le procédé par lequel nous utilisons des expressions mathématiques pour décrire une situation quantitative réelle.

Étape 2 Alimentation : l'opération par laquelle nous affectons des valeurs quantitatives aux paramètres du modèle pour créer un exemple du problème.

Étape 3 Résolution : l'opération qui consiste à trouver une solution à un exemple du problème en appliquant un algorithme donné.

Étape 4 Prise de décision : Le procédé par lequel le scientifique et le décideur permettent d'analyser et d'interpréter les résultats obtenus pour répondre aux questions d'intérêt, et résumer et communiquer la réponse finale.

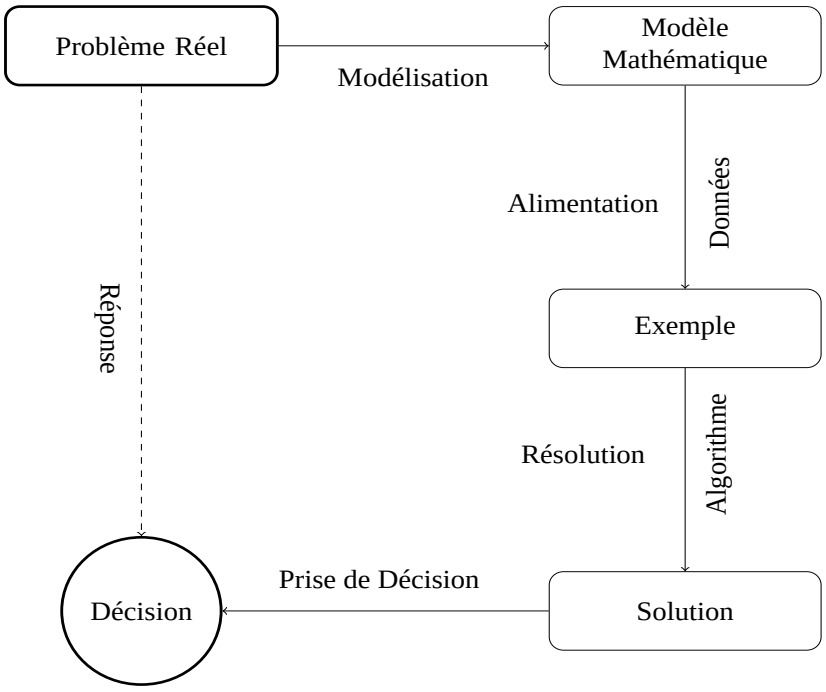


Figure 3.2 – Schéma conceptuel de résolution d'un problème réel.

Chapitre 4

Implémentation et tests numériques

Dans ce chapitre, nous présentons l'implémentation de notre approche de résolution. Nous reportons aussi des résultats numériques obtenus en testant notre code de calcul, sur un exemple réel du problème, et d'utiliser la composante logicielle appelée SOLVEUR et qui est implémentée sous l'application EXCEL.

4.1 Exemple Test

Dans cette section, nous présentons un exemple du problème (CFLP), dont la taille est $n = 15$ clients, et $m = 5$ sites potentiels. Avant de passer le modèle (3.7) au SOLVEUR, nous présentons ici, quelques prétraitements nécessaires.

4.1.1 Matrice de distances (entrepôt-client)

Les valeurs de distance entre l'entrepôt $k \in D$ et le client $j \in C$ sont reportées dans le Tableau (4.1).

4.1.2 Demandes de clients

Dans le Tableau (4.2), la demande de chaque client est reportée.

4.1.3 Couts fixes et capacités des entrepôts

Les valeurs de coût fixe et la capacité de chaque l'entrepôt $k \in D$ sont reportées dans le tableau (4.3).

4.2 Approche de résolution

Dans cette section, nous présentons notre approche de résolution que nous avons adoptée dans ce travail.

4.2.1 Relaxation totale des contraintes d'intégrité

Ici, nous rappelons la relaxation totale des contraintes d'intégrité. Cette relaxation est classique, toute les varibles de décision sont de type réelles. Elle est donnée par,

$$\begin{aligned} & x_{kj} \geq 0, \qquad k \in D,\ j \in C \qquad (d') \\ & y_k \in [0,1]. \quad k \in D \qquad (e') \end{aligned} \tag{4.1}$$

4.2.2 Relaxation partielle des contraintes d'intégrité

Dans ce travail, nous relaxons que la matrice d'affectation $[x_{kj}]$, en d'autre terme, la contrainte d'intégrité n'est imposée que sur la variable de décision y, et dans ce cas, le nombre de variables binaires est m, et l'éspace de recherche engendré et de taille 2^m.

$$\begin{aligned} & x_{kj} \geq 0, \qquad k \in D,\ j \in C \qquad (d'') \\ & y_k \in \{0,1\}. \quad k \in D \qquad (e'') \end{aligned} \tag{4.2}$$

Tableau 4.1 – Coût d'affectation (entrepôt-client)

D\C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	33	25	36	20	27	12	31	32	23	6	16	33	16	47	12
2	24	37	29	8	37	2	17	34	19	20	5	20	17	45	26
3	32	16	31	30	9	29	41	10	19	23	34	42	14	30	15
4	19	31	16	25	23	29	34	9	10	32	34	33	14	17	28
5	4	42	7	15	37	24	20	24	9	35	27	18	17	26	35

Tableau 4.2 – Demande de chaque client

Client	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Demande	4	4	4	5	5	5	5	6	6	6	7	7	8	8	8

Tableau 4.3 – Coût fixe et capacité de chaque entrepôts.

Entrepôt	1	2	3	4	5
Coût fixe	50	50	60	70	70
Capacité	40	40	60	50	50

4.2.3 Coupe plane valide

Le nombre d'entrepots actifs est borné inférieurement comme l'indique l'inégalité suivante qui est une coupe plane valide pour le modèle mathématique (3.7).

$$\sum_{k=1}^{m} y_k \geq \left\lceil \frac{1}{Q_{\max}} \sum_{j=1}^{n} d_j \right\rceil \tag{4.3}$$

4.2.4 Algorithmes

À titre comparatif nous présentons deux algorithmes permettant de résoudre le programme mathématique (3.7).

L'Algorithme (4.1) permet de résoudre notre programme mathématique sans introduire la coupe valide.

Algorithme 4.1 Relaxation partielle sans coupe valide.

1) Résoudre le programme relaxé partiellement (4.2).
2) Si, la matrice $\left[x_{kj}\right]$ est binaire, alors la solution optimale est obtenue. STOP.
3) Sinon, $\exists(k,\ j) \in D \times C$, tel que $0 < x_{kj} < 1$.
   Appliquer l'algorithme de séparation et évaluation (SEP).
4) Aller à l'étape 2)

L'Algorithme (4.2) permet de résoudre notre programme mathématique en introduisant la coupe valide.

Algorithme 4.2 Relaxation partielle avec coupe valide.

1) Résoudre le programme relaxé partiellement (4.2) + la coupe valide (4.3).
2) Si, la matrice $\left[x_{kj}\right]$ est binaire, alors la solution optimale est obtenue. STOP.
3) Sinon, $\exists(k,\ j) \in D \times C$, tel que $0 < x_{kj} < 1$.
   Appliquer l'algorithme de séparation et évaluation (SEP).
4) Aller à l'étape 2)

4.3 Résultats numériques et interprétation

Dans cette section, nous présentons les résultats numériques que nous avons obtenus sur notre exemple.

Pour étudier l'effet de la coupe valide (4.3) sur la résolution du modèle (3.7), nous appliquons les deux algorithmes (4.1) et (4.2). Les résultats obtenus sont représentés par des arborescences dans les deux Figures (4.1) et (4.2).

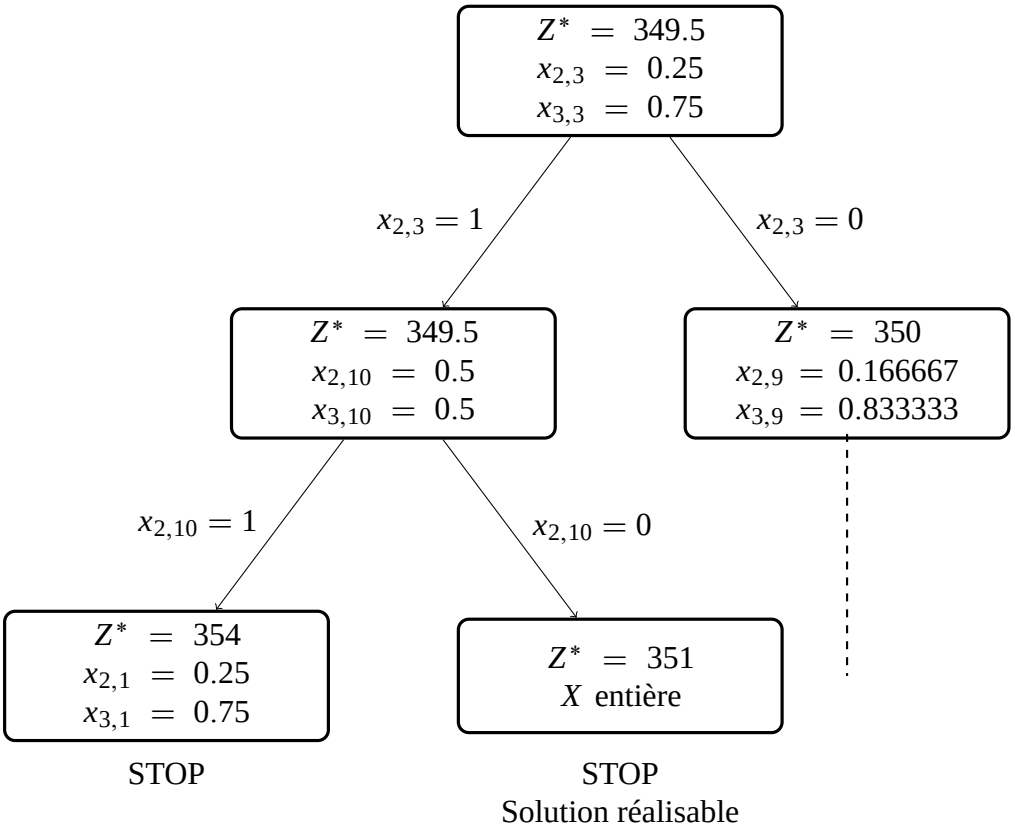


Figure 4.1 – Arborescence sans coupe valide

Durant notre étude, nous avons constaté que de nombreux tests numériques effectués, ont permis d'avérer l'avantages de l'utilisation des coupes valides en conjonction avec un algorithme de séparation et évaluation, pour la résolution du programme mathématique (3.7). Les résultats numériques obtenus affirment l'efficacité de cette approche.

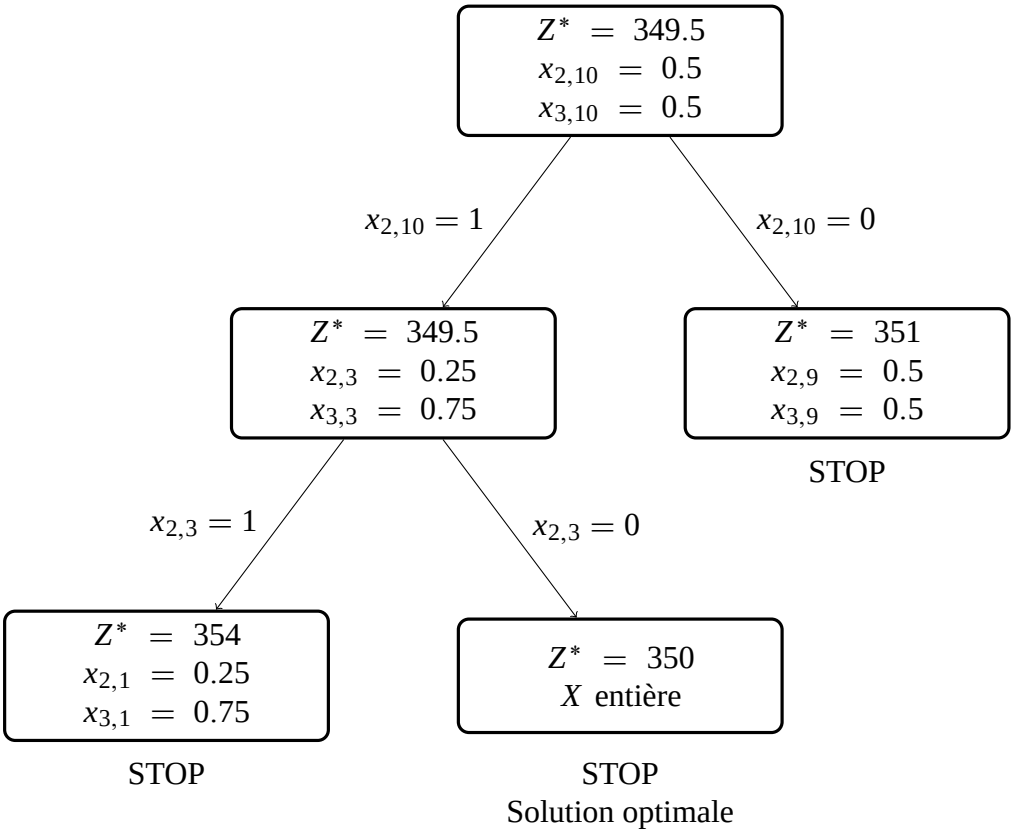


Figure 4.2 – Arborescence avec coupe valide

Conclusion

La rapidité de la prise de décision peut s'avérer primordiale dans la course à la performance exigée par le contexte économique actuel. C'est la raison pour laquelle le gestionnaire ou l'ingénieur cherche toujours à résoudre ses problèmes par des méthodes efficaces et rapides.

Dans ce mémoire, nous nous sommes intéressés au problème de localisation d'entrepôts avec capacité (CFLP). Dans ce problème, un certain nombre entrepôts capacitifs doivent être localisés parmi *m* sites potentiels afin de satisfaire les demandes de *n* clients, en minimisant le coût total d'affectation (client - entrepôt) ainsi que les charges fixes d'établissement de ces entrepôts. Ce problème classé NP-difficile [15] est très étudié dans le domaine de la recherche opérationnelle depuis une dizaine d'années pour ses caractéristiques théoriques et ses applications dans le monde industriel en général. Il est formulé comme un problème de programmation linéaire en nombres binaires (0/1), pour le résoudre, nous avons adopté une approche polyèdrale en appliquant la méthode des coupes planes [14] en conjonction avec un algorithme de Séparation et Évaluation Progressive [15].

Les résultats que nous avons obtenus sur un exemple type, ont montré que l'approche que nous avons adoptée constitue une alternative intéressante pour la résolution du problème (CFLP). L'idée de combiner deux méthodes de résolution, à savoir la méthode de coupe plane avec la méthode de Séparation et Évaluation, s'avère très efficace en pratique, particulièrement pour les exemples de grande taille.

Ces résultats conduisent à envisager quelques perspectives s'articulant autour les points suivants :

- Implémenter une Métaheuristique permettant de résoudre les exemples de grande taille, et de comparer les résultats obtenus avec ceux qui sont obtenus en résolvant le modèle mathématique.
- Proposer une formulation serrée au problème, en pensant d'introduire d'autre type de coupes valides dans le programme mathématique.

Le travail que nous avons effectué dans ce mémoire, nous a permis d'approfondir et de perfectionner nos connaissances théoriques et pratiques dans le domaine de l'optimisation combinatoire et d'acquérir de nouvelles idées en informatique de gestion. Nous espérons que ce travail contribue à l'avancement de la recherche dans ce domaine, et que notre but est atteint.

Notations et abréviations

Symboles :

$\mathbb{R}$	ensemble des nombres réels.
$\mathbb{Z}_+$	ensemble des nombres entiers positives.
$\{0,1\}^n$	ensemble de produit cartésien : $\{0,1\} \times \cdots \times \{0,1\}$, n fois.
$x \in \mathbb{R}^n$	vecteur de taille n : $x = (x_1, x_2, \ldots, x_n)$.
$x^\top$	vecteur transposé de x.
$x \in \mathbb{R}^n_+$	vecteur positif $x_i \geq 0$, $i \in \{1, 2, \ldots, n\}$.
$\sum_{i=1}^n x_i$	la somme : $x_1 + x_2 + \cdots + x_n$.
$\sum_{i=1}^n x_i y_i$	produit scalaire du vecteur x et y.
$\sum_{i=1}^m \sum_{j=1}^n x_{ij}$	$(x_{11} + \cdots + x_{1n}) + (x_{21} + \cdots + x_{2n}) + \cdots + (x_{m1} + \cdots + x_{mn})$.
$\max(z)$; $\min(z)$	maximiser la valeur z ; minimiser la valeur z.

Abréviations :

UFLP	Uncapacitated Facility Location Problem.
CFLP	Capacitated Facility Location Problem.
CP	Cutting-Plane.
B&B	Branch and Bound.
PLNE	Programme linéaire en nombres entiers.
PLNB	Programme linéaire en nombres binaires.

Références

[1] V. Chvatal, Linear Programming. Freeman and Co, 1978.

[2] G. Cohen, Convexité et Optimisation. École Nationale des Ponts et Chaussées et INRIA, 2000.

[3] G. Cohen, Optimisation de grands systèmes. Cours de DEA. Cermics-École nationale des Ponts et Chaussées et INRIA, 2003.

[4] J.C. Culioli, Introduction à l'optimisation, École Nationale Supérieure des Mines de paris, Centre Automatique et Système, Fontainebleau, Ellipse, 1994.

[5] G.B. Dantzing, Linear Programming and Extensions. Princeton. University Press, 1964.

[6] Y. Dodge, S. Gonano-Weber et J.P. Renfer, Optimisation Appliquée, Springer-Verlag France, 2005.

[7] D. Dubois, Modèles mathématiques de l'imprécis et de l'incertain en vue d'application aux techniques d'aide à la décision. Thèse soutenue à l'Université de Grenoble I, 1983.

[8] R. Fourer, D. M. Gay and B. W. Kernigham : A Modeling language for Mathematical Programming. Second Edition, Thomson, USA, 2003.

[9] M. R. Garey, and D. S. Johnson, Computers and intractability : A guide of the theory of NP-completeness. San Fransisco : W.H. freeman and Company 1979.

[10] M. Gondran et M. Minoux, Graphes at algorithms. Eyrolles, 1979.

[11] G. Laporte, Y. Nobert Y and S. Taillefer. Solving a family of multi-depot vehicle routing and location-routing problems. Transportation Science, 22 : 172. 1988.

[12] M. Minoux, Programmation mathématique, Théorie et algorithmes, Collection Technique et Scientifique des Télécommunications, Dunod, Paris, 1983.

[13] J. Nocedal and S. J. Wright, Numerical Optimization, Springer-Verlag, New York, 1999.

[14] M. Sakarovitch, Graphes et Programmation linéaire : Méthodes Mathématiques et Algorithmiques, Programmation Discrète, Hermann, Paris, 1984.

[15] M. Sakarovitch, Optimisation Combinatoire : Méthodes Mathématiques et Algorithmiques, Programmation Discrète, Hermann, Paris, 1984.

[16] P. Toth, D. Vigo, The vehicle routing problem. SIAM, Philadelphia Monograph on Discrete Mathematics and Applications, 2000.