View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by OpenGrey Repository

UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

École Doctorale MIPTIS

Laboratoire d'Informatique (EA6300)

Équipe Ordonnancement et Conduite (ERL CNRS 6305)

THÈSE présenté par :

Xin TANG

soutenue le : 28 novembre 2012

pour obtenir le grade de : Docteur de l'université François - Rabelais

Discipline/ Spécialité : Informatique

Le problème de la sectorisation multicritère en cartographie

THÈSE DIRIGÉE PAR :

SOUKHAL Ameur	Maître de conférence, Université François - Rabelais Tours
T'KINDT Vincent	Professeur, Université François - Rabelais Tours

RAPPORTEURS :

COSTA Marie-Christine	Professeur, ENSTA ParisTech
GANDIBLEUX Xavier	Professeur, Université de Nantes

JURY :

BAIOU Mourad	Chargé de recherche, LIMOS CNRS
COSTA Marie-Christine	Professeur, ENSTA ParisTech
GANDIBLEUX Xavier	Professeur, Université de Nantes
HAMACHER Horst	Professeur, University of Kaiserslautern
SIARRY Patrick	Professeur, Université Paris Est
SOUKHAL Ameur	Maître de conférence, Université François - Rabelais Tours
T'KINDT Vincent	Professeur, Université François - Rabelais Tours

MEMBRE INVITÉ :

BARTHELEMY Jérôme	Manager technique, Articque, Fondettes

Remerciements

Le travail présenté dans cette thèse a été réalisé dans le cadre d'une convention CIFRE au sein de la société Articque à Fondette et de l'équipe Ordonnancement et Conduite du Laboratoire d'Informatique de l'Université de François Rabelais Tours situé à Polytech'Tours. Je tiens donc tout d'abord à remercier l'ensemble des personnes travaillant à Aritcque et à Polytech'Tours qui m'ont accueilli pendant ces trois années.

Je tiens à remercier Georges Antoine STRAUCH qui a permis de débuter ce travail de thèse. Je le remercie également pour m'avoir laisser suffisamment de temps pour la recherche lorsque c'était nécessaire. Merci également à l'ensemble des personnes qui ont travaillé sur ce projet et en particulier à Jérôme Guyon et Florian Sureau.

Je remercie plus particulièrement Vincent T'kindt pour avoir trouvé un sujet de thèse aussi intéressant et pour m'avoir encadré dans la bonne humeur durant ces trois années. Je remercie également Ameur SOUKHAL pour son encadrement et pour m'avoir beaucoup aidé quand le moment est difficile. Je remercie aussi Jean Charles Billaut pour son accueil chaleureux au sein du laboratoire d'informatique. Enfin, je remercie l'ensemble des étudiants qui ont été amenés à travailler sur le projet et en particulier : Tingting Shan, Zhenrui Chu et Claire Merle.

Merci à tous les doctorants, surtout les anciens doctorants du laboratoire et aux autres pour la bonne humeur et les discussions scientifiques que nous avons pu partagées.

Résumé

Les travaux présentés dans cette thèse visent à proposer des méthodes pour résoudre les problèmes de la sectorisation multicritère en cartographie. En premier temps, nous avons défini les problèmes différents de la sectorisation et nous avons établi les liens entre ces problèmes avec les problèmes classiques qui sont bien étudiés dans la littérature : le problème de découpage de district politique, les problèmes de localisation et le problème du partitionnement de graphe.

Deux types de méthodes ont été abordés pour résoudre les problèmes de sectorisation. Des heuristiques ont été développées et elles consistent à calculer un optimum de Pareto pour les différents problèmes. Et pour le problème de sectorisation à partir de pôles, nous avons aussi utilisé et expérimenté un algorithme de boîte pour trouver une représentation du front de pareto.

La méthode exacte *branch and bound* a été utilisée pour résoudre le problème de sectorisation sans pôle prédéfini optimalement. Avant que nous appliquons cette procédure, nous ajoutons quelques inégalités valides dans la formulation mathématique pour restreindre l'espace des solutions et nous développons une procédure de prétraitement pour réduire la taille du problème.

Mots clés : sectorisation cartographique, multicritère, prétraitement

Abstract

The work presented in this thesis aims to propose methods to solve the multicriteria map sectorization problem in cartography. Firstlly, we have defined the different sectorization problems and we have established the links between these problems with some classical problems which are well studied in the literature : political districting problem, location-allocation problems and constrained graph partitioning problems.

Two types of methods have been proposed to solve the sectorization problem. Heuristics have been developed and they compute an optimum Pareto for the different sectorization problems. And for the sectorization problem with predefined centers, we have used a box algorithm and experimented it to find a representation of the Pareto front.

The branch and bound method was used to solve optimally the sectorization problem without predefined centers. Before we apply this procedure, we add some valid inequalities in the mathematical formulation for restrict the space of solutions and we develop a preprocessing procedure to reduce the size of the problem.

Keywords : cartography sectorization, multicriteria, preprocessing

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Contexte et Objectifs

Introduction

1.1 Présentation de l'entreprise Articque

1.1.1 La société Articque

La société Articque, créée en 1989 par Georges-Antoine STRAUCH (GAS), est une PME française. Elle est le leader européen des Systèmes d'Analyse Géographique qui combinent la cartographie et les statistiques pour permettre de visualiser, d'enrichir et de gérer des bases de données cartographique. Ainsi, la société Articque développe des solutions logicielles pour des entreprises et pour des collectivités locales et territoriales (COLAS, LBP, Inter Mutuel Assistance, etc). Connue principalement par le logiciel Cartes & Données et le composant web C&D Web, la société Articque est spécialisée dans le domaine de la cartographie statistique. Avec plus de 5000utilisateurs dans des PME, des grands comptes et des administrations, Articque est un acteur solide du secteur de la cartographie statistique.

La démarche d'innovation et la solidité financière d'Articque ont su convaincre les meilleurs soutiens depuis sa création que ce soit l'aide des chercheurs cartographes du GIP-Reclus [1] ou le soutien financier d'OSEO ANVAR [2], de la Région Centre et du département d'Indre et Loire comme celui de la Communauté Européenne (Label INFO2000).

Articque a obtenu de nombreuses récompenses, le 1er prix des Electrophées dès 1999 ou encore le trophée Cégétel L'Express ou le ch@ppe d'or à l'occasion des Vœux de l'Internet.

Connue au travers de certains de ses sites comme Les Vigies du Littoral, tableau de bord des communes du littoral touchées par l'ERIKA en 1999 ou encore *lafranceelectorale.com* qui a permis à 500 000 internautes de suivre les résultats des élections municipales 2001,

1. Réseau d'étude des changements dans les localisations et les unités spatiales est un groupement d'intérêt public. Il a été créé en 1984 et a fonctionné comme centre scientifique jusqu'en 1997. Il était l'expression d'une ambition de recherche en géographie, et un témoignage des transformations de fond des études géographiques en France.

2. Un grand pôle financier public, créé en 2005 de la fusion de l'ANVAR et de BDpme (Banque de Développement des PME). OSEO est une entreprise publique qui a pour mission de soutenir l'innovation et la croissance des entreprise.

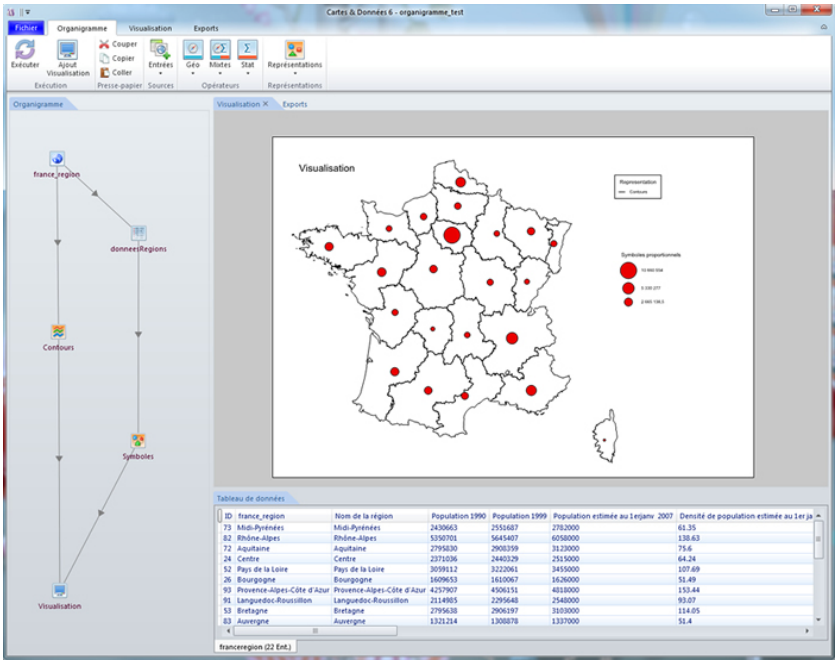

FIGURE 1.1 – Interface de Carte & Données 6

Articque est aujourd'hui une entreprise de 47 personnes en pleine expansion (30% par an), impactée par le dispositif GAZELLE et le Contrat d'Initiative Export mis en place par le Ministère de l'Industrie et le FEDER. Articque affiche de prestigieuses références comme Airbus, Air Liquide, MMA, ADECCO, La Cité de la Musique, La SNCF, la CFE CGC, EDF, la COFACE, CLAAS, Dassault Systemes, Toyota, etc.

Afin de maintenir une qualité irréprochable des produits et applications pour les utilisateurs, Articque dispose d'un comité scientifique dédié. Le comité a un rôle de consultation et d'évaluation des produits et services développés par Articque. Présidé par Hervé Théry, Directeur de recherche au centre de recherche et de documentation sur l'Amérique Latine, et professeur à l'université de São Paulo, le comité scientifique d'Articque regroupe des membres brillants de formations diverses, comme le sont les utilisateurs de Cartes & Données.

1.1.2 Principaux produits de la société Articque

Cartes & Données Cartes & Données est un logiciel de cartographie statistique et décisionnelle pour les décideurs (cf figure 1.1).

Ce logiciel permet de réaliser des cartes en quelques clics et propose des fonctions avancées pour répondre à des besoins plus poussés en matière d'études statistiques et géomarketing. Il permet d'associer des données statistiques à des fonds de cartes pour produire des représentations graphiques claires et très parlantes des analyses. La première version de Cartes & Données est lancée sur Macintosh en 1994. La version actuellement commercialisée est *Cartes & Données 6*.

Le travail de cartographie dans Cartes & Données peut se décomposer en cinq étapes distinctes :

- le choix de la carte géographique
- le choix des données statistiques à considérer
- les traitements statistiques à réaliser pour les analyses statistiques et géomarketing
- le choix de méthodes graphiques pour l'affichage des informations traitées ainsi que les résultats obtenus
- la visualisation du résultat sur la carte.

Dans l'interface de Cartes & Données on retrouve les éléments de la suite bureautique Microsoft Office @ 2010 pour faciliter la navigation dans les différents menus : comme par exemple le ruban qui permet d'accéder à tous les outils regroupés par thèmes. Nous pouvons voir l'affichage des différentes parties de l'interface du logiciel dans la figure 1.1.

En haut de l'interface, tous les outils sont regroupés par thèmes dans le ruban type Microsoft Office @ 2010. Ces outils sont pour réaliser les analyses statistiques et géomarketing.

Sur la gauche de l'interface se situe la fenêtre d'organigramme. Cette fenêtre est l'endroit où tout se passe : nous intégrons des données, des fonds de cartes et nous créons notre analyse au fur et à mesure en reliant des modules les uns aux autres. L'organigramme est le concept clé de Carte & Données. Il offre également la possibilité de communiquer, d'enregistrer la pensée pour l'enrichir par la suite et la partager avec d'autres collaborateurs. L'organigramme est sauvegardé avec la carte et nous pouvons le modifier autant que nous le souhaitons. Grâce à l'organigramme, nous concevons notre analyse directement dans le logiciel Cartes & Données : nous intégrons les données, nous choisissons le fond de carte dont nous avons besoin, nous appliquons le traitement statistique et la représentation adéquats pour visualiser rapidement les résultats sur la carte.

La partie en haut sur la droite de l'interface sont les fenêtres de visualisation et d'export. La fenêtre de visualisation permet de voir en direct le résultat d'analyse en cours dans l'organigramme et de gérer la mise en page complète de document final qui continent les analyses et les cartes. Dans la fenêtre d'export, nous pouvons exporter les résultats d'analyse dans les formats les plus courants. Nous retrouvons également l'outil Multicartes grâce auquel nous automatisons la production de nos cartes en faisant évoluer à la fois le fond de carte et les données, tout en conservant nous choix de mise en page.

La partie en bas sur la droite de l'interface est le tableau de données ou nous pouvons consulter toutes nos données. Nous pouvons trier les données et organiser le travail en direct pour optimiser notre analyse.

Cartes & Données Web Cartes & Données est un viewer cartographique (programme rendant possible la visualisation de l'image sur un écran). Il permet de diffuser facilement les données géospatiales (cartes, graphiques, tableaux, etc) sur un site web (intranet, internet). Il dispose de la puissance de la cartographie statistique et décisionnelle en mode web. Il possède les caractères suivants :

- Diffusion simple et rapide des analyses : cartes, graphiques et tableaux ;
- Interactivité accrue avec les collaborateurs ;
- Rendu esthétique et interface personnalisable ;

- Complémentarité avec Cartes & Données ;
- Navigation intuitive et exports riches ;
- Usage standard ou API d'intégration.

Un des modules le plus important du logiciel Cartes & Données est le *module de sectorisation*. À partir des données statistiques (telles que densités de population et surfaces), ce module permettait de découper une carte géographique en un nombre de secteurs souhaités tout en optimisant un critère, comme par exemple la répartition de la population sur l'ensemble des secteurs doit être équitable. Cependant le résultat obtenu ne permettait pas d'une part, d'obtenir de bonnes solutions et d'autre part de répondre aux attentes des clients de la société qui cherchent un découpage de la carte avec des objectifs multiples (minimiser les distances entre les élément du même secteur tout en équilibrant les charges en terme de densité de la population, parité, superficie, etc.).

Ce travail de thèse a donc pour objectif de palier les quelques déficiences avérées quant à la mise en place d'algorithmes efficaces pour la résolution du problème de sectorisation multicritère. L'apport de ce travail de thèse nous a permis donc de développer la nouvelle version du logiciel Cartes & Données (*Cartes & Données 6*) commercialisée de nos jours (cf chapitre 5).

1.2 Contexte et Champ d'application

Comme on le dit souvent, "le Savoir, c'est le Pouvoir". Pour rationaliser leur activité (ex : être certain mettre les bons commerciaux au bon endroit, ouvrir un nouveau magasin/une nouvelle agence là où il y en a vraiment besoin, mener une politique territoriale/une action là où elle sera utile), les entreprises ou collectivités ont besoin de situer les choses, de localiser les informations qu'elles possèdent, pour prendre en compte la composante géographique dans leurs actions. Le simple tableau Excel ou le graphique "classique" ne le permettent pas, mais la carte, elle, y répond parfaitement. De plus, en localisant les informations, en les replaçant dans leur contexte géographique, on perçoit des choses qu'on ne voit pas habituellement (ex : "c'est normal que ce commercial ne fasse pas un bon résultat de ventes, car les clients qu'il doit visiter sont tous situés très loin de son point de départ !").

Ayant recours à la carte géographique statistique, cela nous permet d'analyser les phénomènes en les croisant : on part de nombreuses données "brutes", on les transforme statistiquement en les croisant entre elles pour calculer des indices, des ratios, des facteurs de corrélation, puis on les superpose sur une carte pour rajouter l'aspect "géographiques" afin d'en tirer des informations utiles qui permettront de prendre les bonnes décisions.

Cette carte est aussi un excellent moyen de communication. Aujourd'hui, les décideurs n'ont plus le temps de parcourir des tableaux et des rapports complexes : ils ont besoin d'informations synthétisées dans des visuels "parlants" dont ils peuvent instantanément extraire les points les plus marquants d'un seul coup d'oeil. La carte, par ce qu'elle utilise des codes visuels (la sémiologie graphique, inventée par Jacques Bertin) adaptés, parce qu'elle synthétise/résume les informations et parce qu'elle permet au lecteur de replacer ces mêmes informations sur un territoire (et donc de rattacher à ce qu'il "lit" sur la carte tout ce qu'il connait déjà par ailleurs de ce même territoire) est donc un vecteur d'information des plus adaptés pour répondre à ces attentes.

C'est dans ce contexte et pour répondre à tous ces objectifs que le travail de thèse s'inscrit. Par nos travaux de recherche, nous devons donc développer des algorithmes et d'améliorer la solution logicielle existante proposée par la société Articque conformément aux exigences suivantes :

- le logiciel doit être simple d'accès : pas besoin de connaitre le SQL ou un quelconque langage de programmation complexe pour l'utiliser. C'est un logiciel qui est plus dans l'esprit "bureautique" (Word, Excel, Access) que "développeur/technicien" (Access, MySQL, R, SPSS, ArcGIS, etc.)
- l'utilisateur "voit" et "comprends" ce qu'il est en train de faire, c'est-à-dire représenter visuellement et graphiquement la réflexion que de l'utilisateur suit pour obtenir le résultat. Autrement dit, faire une carte "parlante" à travers ses outils graphiques (symboles, remplissages, flux, camemberts, histogrammes, etc.)
- le logiciel permet de mener des analyses complexes et abouties à travers les modules développés
- le logiciel permet d'éditer les résultats dans plein de format (PDF, Powerpoint, image, etc.) et même de créer directement un Atlas web dynamique (export Cartes & Données Web) pour une bonne communication.

1.3 Présentation du problème de sectorisation

La cartographie statistique est un domaine d'études en plein essor. Partant d'une carte géographique et de données de nature statistique, la cartographie statistique consiste à réaliser des traitements sur les données et à visualiser les résultats directement sur la carte. Nous avons une grande zone géographique (exemple : département, région, pays, continent, etc), cette zone est composée de plus petits éléments géographiques, dits "*éléments basiques*" (exemple : communes, communautés de communes, départements, etc). Sur une carte géographique, chaque élément basique est représenté en général, par un polygone avec Q ($Q \geq 1$) informations statistiques. Les informations statistiques sont de nature quantitative donc additive, comme la surface, la population, le nombre de clients, etc. Le problème de sectorisation est un problème de cartographie particulier qui consiste à partitionner une zone géographique dans plusieurs plus petites zones (appelées secteurs) en satisfaisant certaines contraintes tout en optimisant certains objectifs. D'un autre point de vue, le problème de sectorisation peut être considéré comme le problème de regroupement d'éléments basiques dans les plus grands groupes géographiques (secteurs). Les secteurs sont acceptables selon les contraintes imposées et les critères prédéfinis. Selon les différentes contraintes et les différents critères, nous pouvons déterminer les classes de problèmes de sectorisation. Nous introduisons par la suite les contraintes et les critères considérés dans notre étude.

1.3.1 Contraintes considérées

Pôle : Un pôle n'est autre qu'un élément basique d'un secteur. Il peut être prédéfini (connu) ou non. Il est à noter que deux pôles ne peuvent pas être dans le même secteur. Dans notre étude, nous abordons trois types de problèmes suivants :

1. **P1** : Pour chaque secteur, le pôle est fixé par le décideur. Ainsi, le nombre de secteurs et les pôles sont connus. Il s'agit donc de chercher une sectorisation à partir de l'ensemble des pôles donnés.
2. **P2** : Les pôle ne sont pas fixés. La sectorisation de la carte géographique détermine donc la localisation de pôle pour chaque secteur.
3. **P3** : Pour chaque secteur, un ou plusieurs pôles candidats sont prédéfinis par le décideur. La sectorisation donne la localisation des pôles retenus parmi ceux présélectionnés.

Le choix de pôles candidats est fixé par le décideur en se basant sur l'importance des valeurs statistiques rattachées aux éléments basiques en question ou sur leur géolocalisation, par exemple.

Nombre de secteurs : Dans notre étude, nous considérons que le nombre de secteurs est connu en avance, noté p. Par définition, c'est le nombre de pôles.

Affectation d'éléments basique : Chaque élément basique doit être affecté exactement dans un seul secteur. Les secteurs définissent une partition de l'ensemble V d'éléments basiques. Nous notons le secteur $j \in \{1, ..., p\}$ par $s_j \subseteq V$, nous avons alors :

$$s_1 \cup ... \cup s_p = V \text{ et } s_j \cap s_k = \emptyset, j \neq k.$$

Connexité Chaque secteur doit être connexe. Afin d'obtenir des secteurs connexe, la relation d'adjacence entre les éléments basiques est nécessaire. Nous pouvons modéliser une carte géographique par un graphe, où les sommets correspondent aux éléments basiques. Nous disons qu'un secteur est connexe, si dans le sous-graphe engendré par les sommets appartenant au secteur concerné, il existe un chemin reliant chaque pair d'éléments du secteur. Les détails sur la modélisation vont être présentés dans la section 1.4.

Homogénéité Les secteurs doivent être homogènes en terme des informations statistiques de chacun. Chaque valeur statistique d'un secteur est définie par la somme des valeurs statistiques correspondantes aux éléments qui lui sont affectés. Formellement, la valeur d'un secteur s_j est donné par : $f_j = \sum_{v_i \in s_j} w_i$, $w_i \in R^Q$ est un vecteur de valuation d'élément v_i. Le vecteur w_i contient Q informations statistiques pour l'élément v_i. Nous avons des contraintes comme, la valeur de chaque secteur ne peut pas passer un seuil (noté par δ) prédéfini par utilisateur.

Forme Concernant la forme de secteur, nous cherchons à conserver au maximum la cohésion géographique des secteurs créés, c'est-à-dire que la forme de chaque secteur ressemble le plus possible à des formes simples et classiques telles que des cercles, carrés ou encore rectangles.

1.3.2 Critères considérées

Comme cela évoqué précédemment, pour une carte géographique, nous disposons de plusieurs données statistiques rattachées à chaque élément basique. Les critères que nous considérons sont basés sur des données quantitatives continues ou quantitatives discrètes. Prenons par exemple, la carte géographique du département de l'Indre-et-Loire (37) où un élément basique correspondant à une commune. On cherche à découper cette carte en p secteurs, autrement dit attacher chaque commune à un secteur. Chaque commune est définie par un ensemble de données quantitatives (mesurables), dites des informations statistiques. Elle peuvent être continues ou discrètes. Comme données discrètes ou discontinues, on peut citer par exemple, taille de la population, nombre d'enfants, nombre de logements sociaux, nombre de places de camping, etc. Quant aux données continues, citons par exemple les revenues des impôts locaux, la superficie, etc. Dans notre étude, nous ne considérons pas des données qualitatives, comme "grand, moyen et petit" pour différencier une commune par rapport à sa taille.

Après l'étude des besoins de clients de la société Articque, les critères identifiés à prendre en compte sont de deux natures : critère d'équilibre et critère d'homogénéité.

- Le critère d'équilibre mesure la différence entre une valeur statistique désirée pour chaque secteur -fixée par le décideur- et la valeur statistique obtenue après sectorisation.
- Le critère d'homogénéité mesure la différence par rapport aux données statistiques considérées entre les valeurs obtenues des différents secteurs.

Pour plus de clarté, regardons le cas de figure suivant où un client désire scinder la région l'Indre-et-Loire en deux parties (secteurs) avec les particularités suivantes : le nombre de clients de chaque secteur est entre intervalle de $-10\%, 10\%$ avec la valeur moyenne de nombre de clients. Cependant, les deux secteurs doivent avoir la même superficie. Le premier souhait du client est vu ici comme un critère d'équilibre quant au second souhait est vu comme un critère d'homoginiété.

Ces deux critères sont à minimiser. Dans notre études, nous allons modéliser ces deux critères sous la forme des type Min-Max et Min-Somme dans la section suivante.

1.4 Modélisation du problème

1.4.1 Transformation d'une carte géographique en un graphe

Une des contraintes difficiles du problème de sectorisation est la *connexité* de secteur. Sur une carte géographique, nous n'avons pas de moyen rapide pour vérifier si un secteur est connexe ou pas avant la visualisation d'une solution sur la carte. Or une modélisation d'une carte géographique par des graphes est un moyen efficace pour vérifier si le résultat obtenu respecte la contrainte de connexité. En effet, les graphes est un outil de modélisation puissant largement étudié. Nous pouvons utiliser cette richesse existante pour nous aider à mieux résoudre le problème de sectorisation.

Un problème de sectorisation peut être modélisé par un graphe $G = (V, E)$ où $V = \{v_1, ..., v_n\}$ est l'ensemble des sommets et E l'ensemble des arêtes (exprimant le voisinage

entre les éléments). Un sommet v_i dans le graphe représente un élément géographique basique sur la carte (ville, par exemple). v_i et v_k sont deux sommets adjacents, reliés par l'arête (v_i, v_k) si ces deux éléments basiques sont géographiquement voisins. Pour chaque sommet v_i, un poids non négatif est associé, noté par le vecteur $w_i \in R^Q, i = 1, ..., n$, où Q représente les valeurs statistiques du $i$ème élément à considérer. Ainsi, les Q valeurs statistiques d'un secteur donné sont définies par la somme des Q valeurs statistiques des éléments définissant le secteur en question.

La figure 1.2 montre la modélisation de la carte de France par un graphe. Dans ce graphe, chaque sommet représente une région, c'est-à-dire chaque élément basique correspond à une région de la France. Nous avons donc 22 éléments. Deux régions sont dites voisines si elles ont une frontière commune, autrement dit, les polygones qui représentent les éléments basiques ont un même segment. Par exemple, les régions Aquitaine (code 72) et Midi-Pyrénées (code 73) sont géographiquement voisines. Dans le graphe obtenu, on relie les sommets 72 et 73 avec une arête. Le cas d'une région isolée, c'est-à-dire ne partageant pas une frontière commune avec une autre région comme le cas d'une île (la Corse, code 94), nécessite un prétraitement pour respecter la contrainte de connexité du secteur susceptible de contenir cet élément isolé. Le prétraitement est défini comme suit : pour chaque région "proche" du sommet isolé, nous la relions à cet élément isolé. Dans l'exemple de la figure 1.2, la région Corse est relié aux trois régions suivantes : Provence-Alpes-Côte d'Azur (code 93), Rhône-Alpes (code 82) et Languedoc-Roussillon (code 91). La notion de "proche" s'exprime en terme de distance séparant l'élément isolé des trois plus proches régions. C'est une contrainte imposé par la société Articque.

1.4.2 Définition des critères

Comme nous l'avons vu dans la section précédente, deux critères à minimiser sont considérés : critère d'*équilibre* et critère d'*homogénéité*. Considérant la $q$ème donnée statistique ($q \in \{1, ..., Q\}$), le critère d'équilibre mesure la différence entre la valeur statistique obtenue et la valeur désirée pour chaque secteur. La valeur désirée (valeur cible) pour un secteur s_j selon q, notée e_j^q, est une valeur prédéfinie par l'utilisateur. La valeur obtenue pour un secteur s_j selon q, notée f_j^q, est la somme de valeurs statistiques des éléments qui sont dans ce secteur. Nous avons donc : $f_j^q = \sum_{v_i \in s_j} w_i^q$. Nous pouvons mesurer cette différence entre la valeur obtenue et la valeur désirée du secteur s_j selon q par $|f_j^q - e_j^q|$. Ainsi, l'objectif est de minimiser cet écart entre les différents secteurs.

Le deuxième critère considéré est le critère d'homogénéité qui mesure la différence selon la donnée q entre les valeurs obtenues de secteurs différents. De la même façon, nous pouvons mesurer la différence entre les valeurs obtenues de deux secteurs différents j et k selon q par $|f_j^q - f_k^q|$.

Dans notre étude, deux familles de critère de types *Min-Max* (minimiser la différence maximale) et *Min-Somme* (minimiser la somme des écarts) sont considérées. Ainsi, sur Q données statistiques, nous avons les quatre familles de critères à minimiser suivants :

$$(i) z_1^q = max_j |f_j^q - e_j^q|, \ q \in Q$$

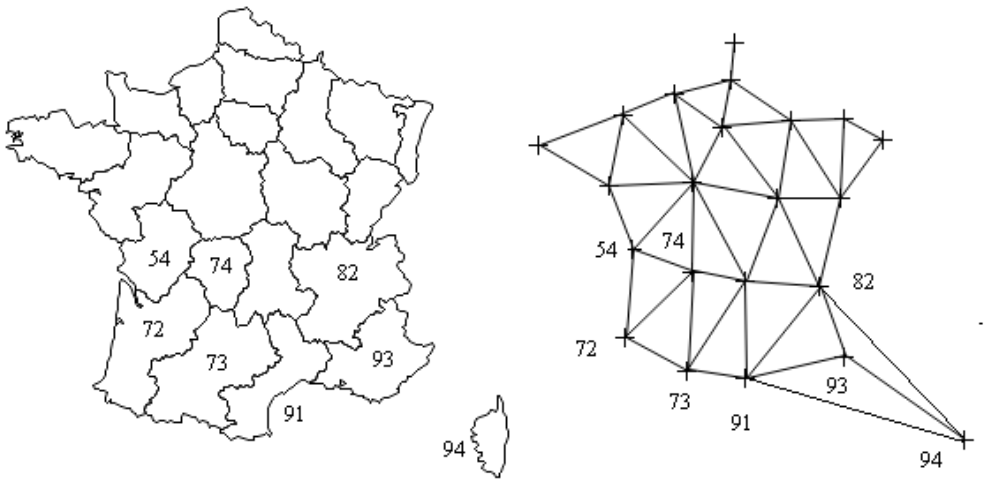

FIGURE 1.2 – Exemple de transformation d'une carte géographique en un graphe

$$(ii) z_2^q = \sum_j |f_j^q - e_j^q|,\ q \in Q$$

$$(iii) z_3^q = max_{j<k} |f_j^q - f_k^q|,\ q \in Q$$

$$(iv) z_4^q = \sum_{j<k} |f_j^q - f_k^q|,\ q \in Q$$

Pour une données statistique q, nous avons : le premier critère z_1^q mesure la différence maximale entre la valeur obtenue et la valeur désirée e_j^q parmi l'ensemble des secteurs j ; le deuxième critère z_2^q mesure la somme de différences entre la valeur obtenue et la valeur désirée de chaque secteur ; le troisième critère z_3^q mesure l'écart maximal entre tout couple de secteurs selon les valeurs statistiques obtenues ; le quatrième critère z_4^q mesure la somme des écarts des valeurs obtenues de tout couple de secteurs.

Après discussion, le choix prioritaire de la société Articque s'est porté sur les critères de type Min-Max, c'est-à-dire z_1^q et z_3^q. Il s'agit donc d'un problème de localisation multicritère et donc d'un problème d'optimisation multiobjectif. Dans la suite, nous présenterons les concepts fondamentaux des problèmes d'optimisation multicritère et les approches de résolution considérées.

1.5 Optimisation multicritère

Comme nous l'avons vu dans la section précédente, la modélisation d'un problème d'optimisation intégrant un seul critère n'est pas toujours suffisante. Par exemple, si on cherche à constituer deux secteurs avec les objectifs suivants : le premier doit couvrir le double de clients que le deuxième secteur, et en même temps, les deux secteurs doivent avoir quasiment la même superficie. Ce type de problème que la société Articque rencontre et doit donc disposer d'un outil qui permet au décideur de faire son choix.

Dans la suie, nous allons rappeler quelque définitions nécessaires pour faciliter la lecture de ce document. Il ne s'agit pas de reprendre toute la théorie de l'optimisation multicritère. Nous renvoyons le lecteur intéressé aux ouvrages suivants [Ehrgott, 2005], [T'kindt et Billaut, 2006] et [Collette et Siarry, 2002] pour un état de l'art complet sur les problèmes d'optimisation multicritère.

1.5.1 Problème d'optimisation multicritère

Soit x une solution d'un problème d'optimisation multicritère Π. Notons $Z(x)$ avec $Z(x) \in R^K$ un vecteur de critère correspondant à la solution x, K est le nombre de critères à optimiser du problème Π. Lors qu'il s'agit d'un problème de minimisation, nous cherchons donc à minimiser les fonctions de coûts. Dans le cas contraire, il s'agit de maximiser les gains. Sans perte de la généralité, nous supposons que les fonctions à optimiser sont à minimiser.

Mathématiquement parlant, un problème d'optimisation multicritère se présente sous la forme suivante :

$$\text{Minimiser } Z(x) \text{ avec } Z(x) = [Z_1(x); ...; Z_K(x)]^T$$

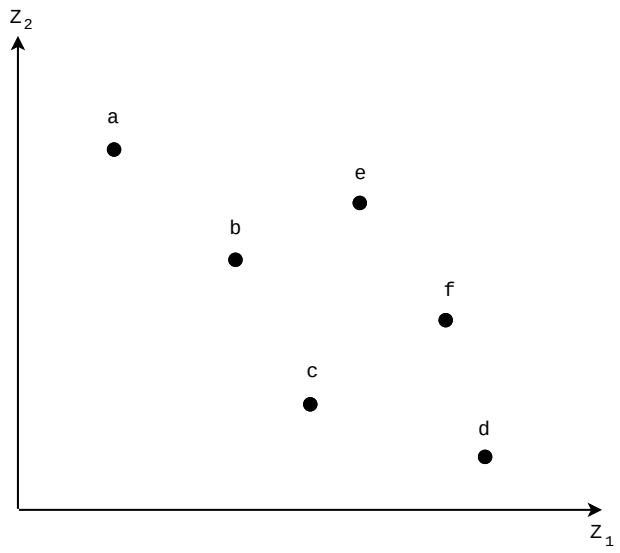

FIGURE 1.3 – Représentation des solutions S dans l'espace des critères

sous contraintes

$$x \in S \text{ avec } S = \{x/[g_1(x); ...; g_M(x)]^T \leq 0\}$$

Notons S l'ensemble des solutions réalisables qui respectent les M contraintes dans l'espace des K critères. Lorsque le nombre de critères K est égal à deux, il est possible de représenter l'ensemble S sur un repère où les axes des abscisses et des ordonnées correspondent à chacun des deux critères, comme sur la figure 1.3. Parmi les solutions S, certaines solutions sont plus intéressantes que d'autres. Par exemple, sur la figure 1.3, la solution e est moins bonne que la solution b pour chacun des deux critères. Cependant, si on considère les deux solutions a et b, nous remarquons que a est meilleure que b sur le critère Z_1, mais b est meilleure que a sur le critère Z_2. Si on considère les deux critères simultanément, il n'est pas possible de déterminer qui est meilleure entre a et b.

La notion d'optimalité dans un contexte d'optimisation multicritère est différente de celle dans un contexte monocritère. Ainsi, pour comparer deux solutions dans un contexte multicritère, nous nous référons à la relation de dominance de Pareto qui est définie comme suit :

Définition 1 *[T'kindt et Billaut, 2006] On dit que la solution x_2 est Pareto-dominée par la solution x_1 si : x_1 est au moins aussi bon que x_2 dans tous les critères, et x_1 est strictement meilleur que x_2 dans au moins un critère.*

Par exemple, sur la figure 1.3, les solutions e et f sont Pareto-dominées par la solution c.

Les solutions qui dominent les autres mais ne se dominent pas entre elles sont appelées solutions optimales au sens de Pareto (ou solutions non-dominées). L'ensemble des solutions optimales Pareto d'un problème d'optimisation multicritère est également appelé front de

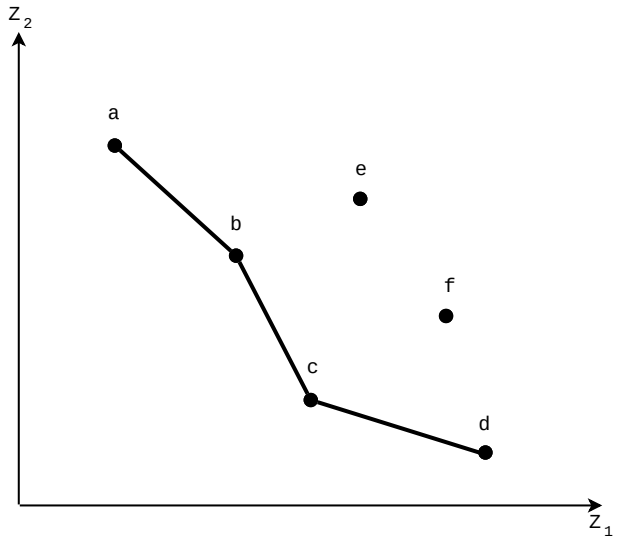

FIGURE 1.4 – Représentation du front de Pareto

Pareto et est représenté sur la figure 1.4. Parmi les optimums de Pareto, une distinction est souvent faite entre les solutions appartenant ou non à l'enveloppe convexe du front de Pareto dans l'espace des critères :

- les solutions dites supportées qui sont sur l'enveloppe convexe. Selon l'exemple de la figure 1.4, a, c et d sont des solutions supportées.
- les solutions dites non-supportées qui ne sont pas sur l'enveloppe convexe. Selon l'exemple de la figure 1.4, la solution b est non-supportée.

Cette distinction existe car il s'agit de deux catégories différentes de solutions. Une solution supportée x peut être obtenue en optimisant une combinaison linéaire des critères $Z_\ell = \sum_{q=1}^{K} \lambda^q Z_q$, telle que $\sum_{q=1}^{K} \lambda^q = 1$ en choisissant la bonne valeur de λ. Cela revient à transformer le problème multicritère en problème monocritère.

Parmi tous les optima de Pareto, il peut être intéressant de connaître les valeurs minimales et maximales de chacun des critères. Pour cela les points *idéal* x_I et *nadir* x_N sont définis comme sur la figure 1.5 pour borner les valeurs des critères.

Définition 2 *Les coordonnées du point idéal sont obtenues en optimisant chaque fonction de critère séparément.*

Définition 3 *Les coordonnées du point nadir correspondent aux pires valeurs obtenues par chaque fonction de critère lorsque l'on restreint l'espace des solutions au front de Pareto.*

Le point idéal est utilisé dans beaucoup de méthodes d'optimisation multicritère comme point de référence. Le point nadir sert à restreindre l'espace de recherche, il est utilisé dans certaines méthodes d'optimisation multicritère [T'kindt et Billaut, 2006].

Pour un même problème d'optimisation multicritère, toutes les représentations du front de Pareto ne sont pas équivalentes. En effet, la représentation idéal du front de Pareto devra

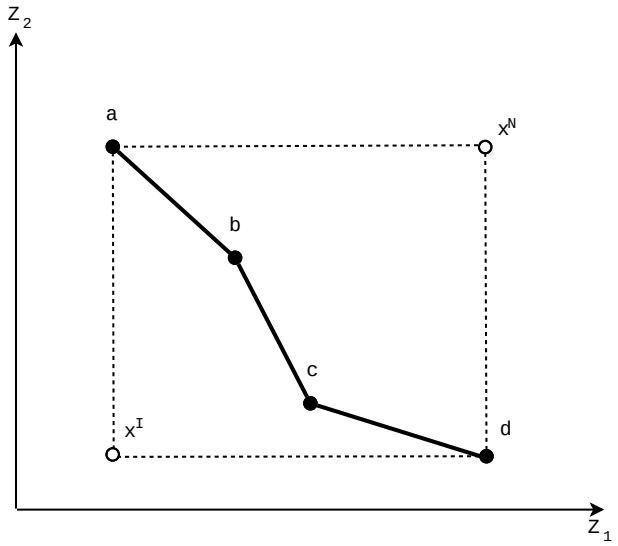

FIGURE 1.5 – Représentation du point idéal et du point nadir

être constituée de points solution de notre problème répartis de manière uniforme sur le front de Pareto (voir la figure 1.6).

Selon la figure 1.6.a, les points représentant le front de Pareto ne sont pas répartis de manière uniforme. En effet, si l'utilisateur rejette une solution parce qu'elle ne convient pas, alors il doit choisir une autre parmi celles proposées. Dans ce cas de figure, le décideur sera confronté à choisir entre deux critères (bon pour un, mais mauvais pour l'autre) sans vraiment avoir une solution de compromis. Ainsi, passer d'une solution à l'autre conduit à une différence brutale sur les valeurs des critères considérés.

La détermination d'une bonne représentation du front de Pareto sera un critère de choix d'une méthode d'optimisation multicritère.

1.5.2 Méthodes de détermination des optima de Pareto

Les optima de Pareto correspondent à des solutions de "meilleur compromis" entre différents critères conflictuels. C'est l'utilisateur (le Décideur) qui peut choisir parmi l'ensemble des optima de Pareto la solution qui est la plus satisfaisante pour son problème. Pour calculer les optima de Pareto pour les problèmes d'optimisation multicritère, il existes différentes approches. Il est possible de classer ces méthodes en fonctions des informations fournies par le Décideur. Le lecteur intéressé peut se reporter à [T'kindt et Billaut, 2006]. Néanmoins, rappelons l'approche par combinaison linéaire et l'approche ε-contrainte.

La méthode de pondération des fonctions de critère

Dite approche par combinaison linéaire, c'est une méthode plus évidente pour calculer les optima de Pareto. Elle est d'ailleurs appelée l'"approche naïve". Le but est de rever-

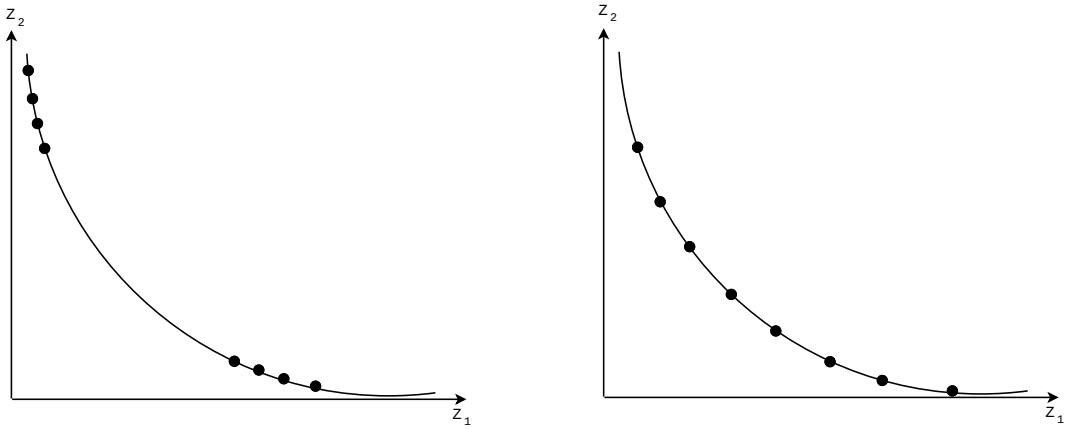

FIGURE 1.6 – Deux représentations différentes du front de Pareto

nir à un problème d'optimisation monocritère, dont il existe de nombreuses méthodes de résolution. La manière la plus simple est d'associer à chaque critère un coefficient (poids). Ainsi, l'objectif global correspond à faire la somme des valeurs pondérées des critères. Le problème d'origine d'optimisation multicritère peut s'écrire comme suit :

$$\text{Minimiser } Z(x) = \sum_{i=1}^{K} w_i Z_i(x)$$

sous contrainte

$$x \in S \text{ avec } S = \{x/[g_1(x); ...; g_M(x)]^T \leq 0\}$$

Fréquemment, les coefficients de pondération respectent la relation :

$$\sum_{i=1}^{K} w_i = 1 \text{ et } 0 \leq w_i \leq 1 \ , \forall i \in \{1, ..., K\}$$

Cette méthode est l'une des première utilisées. D'un point de vue algorithmique, elle est très efficace. On peut, en faisant varier les coefficients de pondération, retrouver la surface de compromis, si le domaine réalisable est convexe.

L'approche ε-contrainte

L'approche ϵ-contrainte consiste à minimiser un seul critères en bornant supérieurement les autre $K-1$ critères. La démarche est la suivante :

- on choisit un critère $k \in \{1, ..., K\}$ à minimiser (prioritairement) ;
- on choisit un vecteur de contraintes $\epsilon_i, i \in \{1, ..., K\}, i \neq k$, qui n'est autre qu'une borne supérieure pour les autres critères ;
- on transforme le problème multicritère en monocritère où la solution calculée doit satisfaire les $K-1$ nouvelles contraintes d'inégalité portées sur les autres critères.

Le problème d'optimisation multicritère d'origine peut être ainsi écrit comme suit :

$$\text{Minimiser } Z_k(x)$$

sous contrainte

$$Z_i(x) \leq \epsilon_i \ \forall i \in \{1, ..., K\}, i \neq k$$
$$x \in S \text{ avec } S = \{x/[g_1(x); ...; g_M(x)]^T \leq 0\}$$

L'approche ϵ-contrainte est largement employée dans la littérature car elle es facile à utilisé dans un algorithme interactif : le Décideur peut spécifier et modifier interactivement les bornes et analyser l'influences de ces modifications sur la solution finale. Un autre avantage de l'approche ε-contrainte réside dans le fait qu'à chaque itération on se ramène à un problème monocritère pour lequel il se peut que nous disposons d'un algorithme de résolution efficace.

1.5.3 Champs d'études de la thèse

Dans la Section 1.4.2, nous avons mentionné les deux types de critères Min-Max considérés pour l'étude du problème de sectorisation : le critère d'équilibre et le critère d'homogénéité, notés z_1 et z_3.

Quand à l'approche de résolution retenue est l'approche ε-contrainte pour calculer un optima de Pareto du problème bicritère avec z_1 et z_3. Nous prenons le critère z_1 comme un critère à minimiser tout en bornant supérieurement le critère z_3 par ε donnée par le Décideur :

$$\text{Minimiser } z_1^{q_1} = max_j |f_j^{q_1} - e_j^{q_1}|$$

sous contrainte

$$z_3^{q_2} = max_{j<k} |f_j^{q_2} - f_k^{q_2}| \leq \epsilon$$
$$\text{avec } q_1, q_2 \in \{1, ..., Q\} \ , \ q_1 \neq q_2$$

1.6 Conclusion

Dans ce chapitre nous avons présenté le problème de sectorisation multicritère en cartographie statistique. Nous avons aussi introduit les contraintes du problème et les critères considérés. En s'appuyant sur la théorie des graphes, nous représentons une carte géographique par un graphe planaire.

Comme nous le constatons, il s'agit d'un problème de localisation bicritère. Le chapitre suivant propose une revue de la littérature rapide sur le problème de localisation et ses diverses versions où nous nous focaliserons principalement sur les problèmes qui nous ont semblé proche du sujet faisant objet de notre étude, et qui ont été sources de nos inspirations quant aux approches de résolutions proposées tels que le problème de p-médian, le p-centre ou encore le problème de recouvrement.

Chapitre 2

État de l'art

Le problème de sectorisation est dans un problème à l'intersection de plusieurs domaines de recherche. Il existe plusieurs types de problèmes qui ont des liens avec notre problème. Ce chapitre présente un état de l'art des différents problèmes qui ressemblent au problème de sectorisation et qui ont été étudiés dans la littérature : les problèmes de découpage de districts politiques, les problèmes de localisation, les problèmes de partitionnement de graphe.

2.1 Les problèmes de découpage de districts politiques

Dans les problèmes de découpage de districts politiques, l'objectif est de découper une grande zone géographique, comme un état ou une ville, en sous-zones (districts) à partir de laquelle les candidats politiques sont élus. Ce problème est particulièrement important dans les démocraties où chaque district élit un seul membre à une assemblée parlementaire. C'est par exemple le cas au Canada, de la plupart des états aux États-Unis et en Nouvelle-Zélande. En général, le processus de redécoupage doit être effectué périodiquement afin de tenir compte des déplacements de la population. La longueur de ces périodes varie d'un pays à l'autre : par exemple, en Nouvelle-Zélande tous les cinq ans, mais au Canada et aux États-Unis tous les dix ans. Pour faciliter ce processus, des chercheurs ont développé de nombreuses procédures automatique et neutres. Pour en savoir plus sur les problèmes de découpage de districts politiques, le lecteur est renvoyé à [Grilli di Cortona *et al.*, 1999] et [Bozkaya *et al.*, 2005].

Dans le passé, le découpage de district politique a été manipulé en visant à favoriser certains partis politiques ou en favorisant la discrimination envers les minorités sociales ou ethniques. Un cas célèbre est arrivé dans l'État du Massachusetts dans le début du 19ème siècle. Le gouvernement de l'état a proposé un district électoral sous la forme d'une salamandre afin de gagner un avantage électoral. Le gouverneur de l'état à cette époque était Elbridge Gerry et cette pratique est devenue connue sous le nom de gerrymandering (charcutage électoral).

Pour éviter toute interférence politique dans le processus de découpage de districts, de nombreux états aux aux États-Unis ont mis en place une commission neutre, dont les

fonctions comprennent l'élaboration de districts politiques qui satisfont un certain nombre de critères. Par exemple, certaines législatures imposent que les districts doivent être contiguë et que leurs populations qui ont le droit de vote doivent se situer dans un intervalle donné. Un autre critère important est que les districts doivent être compacts pour éviter les soupçons de gerrymandering. Nous présenterons ci-dessous les critères les plus utilisés dans la littérature.

Égalité de population. Lors de la conception des districts électoraux, le critère principal est l'équité pour respecter le principe "un homme-une voix", c'est-à-dire que chaque vote a le même pouvoir, afin d'arriver à une présence équitable d'élus entre partis. Toutefois, selon le pays, une déviation par rapport à la cible (population égale) est autorisée pour prendre d'autres critères en compte. Cette déviation admise par rapport à la population moyenne varie de 5% en Nouvelle-Zélande à 50% au Canada. Aux États-Unis, cependant, l'égalité de la population a été jugé très important et la déviation actuellement autorisée est dans la plupart des cas de moins de 1%.

Compacité. Un district est dit être géographiquement compacte si sa forme est ronde et non perturbée. La compacité est un concept très intuitif et une définition rigoureuse de compacité n'existe pas. En fait, aux États-Unis, la compacité a été défini tout simplement comme l'absence de gerrymandering.

Contiguïté. Intuitivement, un district est contigu, s'il est possible de marcher à partir de chaque point dans le district à chaque autre point sans avoir traversé un autre district. La première raison est de protéger encore une fois contre l'effet gerrymandering et la seconde raison est tout simplement pour des raisons administratives.

Conformité administrative. Dans de nombreux cas, les districts nécessitent de prendre en compte des frontières administratives déjà existantes, comme des villes ou des régions. [Ricca et Simeone, 2008] adoptent un indice de conformité administrative qui est défini sur la base du nombre d'unités territoriales qui causent la divergence entre les districts administratifs et les districts électoraux. Ces unités sont appelées unités *mal placés*. Plus grand le nombre de ces unités, et pire est la valeur résultant de l'indice de conformité administrative.

Autres critères. D'autres critères couramment utilisés sont les suivants : probabilité égale de représentation afin d'assurer que les électeurs minoritaires aient le même poids que les autres électeurs ; homogénéité socio-économique pour assurer une meilleure représentation des résidents qui partagent des mêmes préoccupations ou points de vue ; le respect de l'intégrité des communautés, c'est-à-dire, éviter le fractionnement des communautés entre plusieurs districts.

La littérature scientifique sur le découpage de districts politiques montre que il n'y a pas de consensus sur le ou les critères légitimes et sur la façon de mesurer ces critères.

Depuis les début des années 1960, plusieurs méthodes approchées et exactes ont été proposées pour le problème de découpage de districts politiques. Ces méthodes sont basées sur l'une des deux formulations en programmation linéaire en nombres entiers du problème. La première approche de programmation mathématique a été proposée [Hess *et al.*, 1965] qui formulent le problème comme un problème de localisation-allocation. Soit I l'ensemble des unités territoriales, et soit J l'ensemble des unités territoriales utilisées comme "centre" de district potentiel : ainsi on fait l'amalgame entre le numéro de district et le numéro de l'unité qui lui est affecté comme centre" ($J \subseteq I$). De même, le nombre de districts à créer est donné et est noté p avec $p \leq |J|$. Le coût c_{ij} d'affectation d'une unité i à un "centre" j (district j) est une fonction de la distance euclidienne entre l'unité i et le "centre" j. La population de l'unité i est égale à p_i et la population d'un district doit se situer dans un intervalle $[a, b]$. Soit x_{ij} une variable binaire égale à 1 si et seulement si l'unité i est affectée au "centre" j (district j). Le problème est modélisé comme un problème de p-médian avec capacité comme suit.

$$\text{Min} \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{2.1}$$

sous contraintes

$$\sum_{j \in J} x_{ij} = 1, \quad \forall i \in I \tag{2.2}$$

$$\sum_{j \in J} x_{jj} = p, \tag{2.3}$$

$$x_{ij} \leq x_{jj}, \quad \forall i \in I, \quad \forall j \in J \tag{2.4}$$

$$a \leq \sum_{i \in I} p_i x_{ij} \leq b, \quad \forall j \in J \tag{2.5}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in I, \quad \forall j \in J \tag{2.6}$$

Dans cette formulation, la fonction objectif (2.1) mesure la compacité, tandis que l'égalité de population est prise en compte par contraintes (2.5). Les contraintes (2.2) assurent que chaque unité territoriale est affectée à un district, et le nombre de districts est égal à p par la contrainte (2.3). Par les contraintes (2.4), une unité territoriale ne peut pas être affectée au numéro de district dont le "centre" n'a pas été sélectionné. Il est important de noter que cette formulation ne garantit pas la contiguïté des districts, bien que ce sera favorisé par la fonction objectif.

En général, la solution optimale issue de cette formulation n'est pas une solution appropriée pour le problème de découpage de districts politiques, mais cette formulation peut servir comme un guide pour une méthode approchée de localisation-allocation. Dans cette approche, les phases de localisation et d'allocation sont décomposées. Dans la phase de localisation, les centres des districts sont choisis, et dans la phase d'allocation les unités territoriales sont affectées à ces centres. Cette approche est utilisée par différents auteurs qui proposent des heuristiques. [Hess *et al.*, 1965] proposent une heuristique en 4 étapes : (1) commencer avec un ensemble arbitraire de centres de districts, (2) résoudre le problème de transport associé à l'allocation des unités territoriales vers les centres, où la demande pour chaque unité territoriale est la population d'électeurs, la charge (l'offre) de chaque centre est la quantité de la population de chaque district, et la distance entre un centre et

une unité territoriale est le carré de la distance euclidienne entre leurs centres de gravité, (3) affecter chaque unité territoriale qui est fractionné au centre fournissant la plus grande fraction de sa demande, et (4) calculer le centre de gravité de chaque district dérivés et résoudre le problème de transport avec ces nouveaux centres jusqu'à la convergence des centres. [George *et al.*, 1997] résolvent ce problème jusqu'à 35000 unités territoriales en relâchant simplement les contraintes d'intégralité sur les variables d'affectation. Toutefois, la résolution du problème relaxé donne des solutions optimales qui répondent à l'égalité de population mais des unités sont souvent affectées à plusieurs centres (districts). Fleischmann et Paraschis [Fleischmann et Paraschis, 1988] utilisent une heuristique similaire à celle de Hess. Pour déterminer à quels centres sont affectés les unités fractionnées, ils proposent un algorithme qui maximise le nombre d'unités fractionnés pouvant être affectées sans que la contrainte (2.5) du modèle précédent ne soit violée. Cependant, dans cette approche, les unités fractionnées ne peuvent pas toutes être résolues automatiquement. [Hojati, 1996] propose l'utilisation de la relaxation lagrangienne afin de déterminer les centres des districts. Il montre que le choix des centres a un impact important sur les districts résultant. [Kalcsics *et al.*, 2010] utilisent une règle particulière d'allocation des centres. Ils affectent les unités territoriales aux centres en prenant compte les distances, c'est-à-dire qu'une unité territoriale est affectée au centre le plus proche. Cette façon d'allouer peut générer des districts compacts et souvent contigus. Néanmoins, le découpage est moins bon sur le critère de l'égalité de population.

Dans la deuxième formulation de programmation mathématique, I est l'ensemble des unités territoriales comme dans la première formulation et J est l'ensemble des districts faisables. Un indice binaire a_{ij} est égale à 1 si et seulement si l'unité i appartient au district j. Un coût c_j est affecté au district j et le nombre de districts est toujours égal à p avec $p \leq |J|$. La variable binaire x_j prend la valeur 1 si et seulement si le district j est choisi. Cette formulation proposée par [Garfinkel et Nemhauser, 1970] est comme suit :

$$\text{Min } \max_{j \in J} c_j x_j \tag{2.7}$$

sous contraintes

$$\sum_{j \in J} a_{ij} x_j = 1, \quad \forall i \in I \tag{2.8}$$

$$\sum_{j \in J} x_j = p, \tag{2.9}$$

$$x_j \in \{0, 1\}, \quad \forall j \in J \tag{2.10}$$

Partant de cette formulation, ils proposent une méthode exacte en deux phases : dans la première phase, tous districts faisables (candidats), où un district faisable indique qu'il satisfait les contraintes de contiguïté, de compacité et d'égalité de population ; dans la deuxième phase, les districts sont sélectionnés à partir de l'ensemble des candidats qui peuvent couvrir chaque unité territoriale exactement une fois, et qui minimisent le coût maximale engendré par les districts. [Mehrotra *et al.*, 1998] reprennent ce modèle, et considèrent la compacité comme la fonction objectif. Ils développent un algorithme de génération de colonnes et ils envisagent beaucoup plus de district faisables que l'approche initiale de Garfinkel et Nemhauser. En effet, dans leur modèle, la faisabilité d'un district n'est basée que sur la contiguïté et l'égalité de population. Leur méthode reste une heuristique parce

que, premièrement, l'approche qu'ils utilisent pour garantir la contiguïté des districts n'est pas exact. De plus, le sous-problème dans lequel les nouvelles colonnes sont générées est NP-difficile et n'est pas résolu optimalement.

Les métaheuristiques sont aussi utilisées pour le problème de découpage de districts politiques. Ce sont des méthodes heuristiques qui tiennent à améliorer une solution existante en échangeant successivement des unités territoriales entre districts voisins pour minimiser la valeur de la fonction objectif. L'avantage de ces méthodes, par rapport à des méthodes exactes, qu'elles permettent de résoudre des problèmes où la fonction objectif n'est pas linéaire, où encore lorsque la contrainte de contiguïté est imposée. [Bozkaya *et al.*, 2003] considèrent le problème de découpage de districts politiques comme un problème d'optimisation multicritère et ils traitent la contiguïté comme une contrainte : tous autres critères sont agrégés dans la fonction objectif. Ils proposent un algorithme de recherche tabou pour résoudre ce problème. [Ricca et Simeone, 2008] développent une descente locale, une recherche tabou et un recuit simulé pour résoudre ce problème et comparent leurs performances.

Après les études sur les problèmes de découpage de districts politiques, nous remarquons que ce problème et le problème de sectorisation, tous consistent à partitionner une grande zone dans plusieurs plus petites zones connexes. Les différences entre ces deux problèmes se reposent sur leurs contraintes et leurs objectifs. Dans le problème de découpage de districts politiques, nous voulons l'égalité sur la population, et l'égalité est toujours pris en compte comme une contrainte imposée ; mais dans le problème de sectorisation, nous avons un critère d'équilibre qui mesure la différence entre une valeur statistique désirée pour chaque secteur et la valeur statistique obtenue après sectorisation, donc il est plus général que la considération de l'égalité. La deuxième différence est que dans le problème de découpage de districts politiques, nous considérons souvent le compacité comme l'objectif ; mais dans le problème de sectorisation, nous ne le considérons pas comme une contrainte ou un objectif, c'est plutôt une préférence.

2.2 Problèmes de localisation

Dans cette section nous nous intéressons aux problèmes de localisation qui présentent de fortes similitudes avec les problèmes de sectorisation. Après une introduction à ces problèmes dans la section 2.2.1, nous présentons une typologie dans la section 2.2.2. Enfin, nous terminons par un état de l'art sur les problèmes survenant dans le cadre de réseaux dans les sections 2.2.3 et 2.2.4.

2.2.1 Introduction de problèmes de localisation

La plupart des problèmes de localisation peuvent être définis comme suit. Étant donné un espace, une fonction de distance qui est définie entre deux points dans cet espace, un certain nombre de clients existants qui se trouvent dans l'espace et qui ont une certaine demande pour un produit (ou un service), l'objectif est de localiser une ou plusieurs activités dans cet espace pour satisfaire une partie ou la totalité des demandes des clients. Un problème de localisation est défini principalement par : (1) les clients, qui existent déjà

dans certains endroits, (2) les activités, qui vont être localisées, (3) l'espace où les clients et les activités sont localisés, et (4) une métrique qui indique les distances ou les temps de parcours entre les clients et les activités. Dans la suite, nous notons que I l'ensemble des clients, et J l'ensemble de localisation des activités.

2.2.2 Typologie

De nombreuses classifications des modèles de localisation ont été proposées dans la littérature ([Handler et Mirchandani, 1979][Brandeau et Chiu, 1989][Eiselt *et al.*, 1993] [Carrizosa *et al.*, 1995][Hamacher et Nickel, 1998]). [Hamacher et Nickel, 1998] proposent également une notation en 5 champs *Pos1/Pos2/Pos3/Pos4/Pos5*. La signification de chaque champ est décrite comme suit : *Pos1* contient des informations sur le nombre et le type de nouvelles activités ; *Pos2* contient l'espace de décision, cette information peut distinguer entre des modèles dans un espace réel et dans un réseau, des modèles continus et discrets ; *Pos3* décrit des caractéristiques du modèle de localisation, telles que des informations sur la faisabilité de solution, la restriction sur capacité, etc. *Pos4* indique la relation entre les activités nouvelles et existantes. Cette relation peut être exprimée par une fonction de distance ou par des coûts affectés. *Pos5* est la description de la fonction objectif. Cette notation est peu utilisée dans la littérature. Nous nous intéressons maintenant à la classification proposée par [Eiselt et Sandblom, 2004]. Cette classification est basée sur une décomposition des principales caractéristiques des problèmes de localisation. Cette classification est communément utilisée par la communauté de localisation. Dans la suite de section nous présentons les différents critères de classification utilisées par Eiselt et Sandblom pour distinguer les problèmes de localisation.

2.2.2.1 Espace et distance

Nous utilisons souvent l'espace dans lequel les activités sont localisées pour classifier les problèmes de localisation. Nous distinguons entre les problèmes de localisation dans un espace réel de dimension d et dans un réseau. Ceux-ci peuvent être subdivisés en problèmes continus ou discrets. Dans des problèmes continus, les activités à localiser peuvent généralement être placées n'importe où sur le plan ou le réseau. En revanche, dans des problèmes discrets, les activités ne peuvent être placées que dans un nombre limité de points admissibles.

En plus de l'espace, nous avons également besoin d'une mesure pour les distances entre paires de points. La distance ou certaine mesure liée avec la distance (le temps ou le coût) est un élément fondamental dans les problèmes de localisation. Les fonctions de distance les plus populaires sont les métriques et les jauges. Plus particulièrement, pour des problèmes planaires, les fonctions de distance les plus utilisées sont différentes versions de la distance de Minkowski l_p. Selon cette métrique, la distance entre deux points (x_i, y_i) et (x_j, y_j) est définie comme suit à l'aide d'un paramètre p :
a. Métrique de Manhattan, pour $p = 1$,

$$d_{ij}^1 = |x_i - x_j| + |y_i - y_j| \tag{2.11}$$

b. Métrique de euclidienne, pour $p = 2$,

$$d_{ij}^2 = \left\{(x_i - x_j)^2 + (y_i - y_j)^2\right\}^{0.5} \tag{2.12}$$

c. Métrique de Chebyshev, pour $p \to \infty$,

$$d_{ij}^{\infty} = max\{|x_i - x_j|; |y_i - y_j|\} \tag{2.13}$$

D'autres fonctions telles que des jauges sont proposées par différents chercheurs. (voir notamment [Plastria, 1995]).

En revanche, la distances entre deux points dans les problèmes de localisation dans un réseau sont mesurées comme le plus court chemin entre eux. Dans un réseau de n points, il faudra calculer en $O(n^3)$ toutes les distances entre points en utilisant une méthode appropriée, telle que la méthode de Floyd-Warshall.

En général, les problèmes de localisation continus sont souvent liées à l'étude de l'emplacement des activités sur un plan, sont considérés comme des problèmes d'optimisation non linéaire, alors que les problèmes de localisation discrets, qui sont plus souvent des problèmes dans un réseau, implique des variables binaires.

2.2.2.2 Nombre d'activités à localiser

Un autre critère pour classer les problèmes de localisation est le nombre d'activités à localiser. Dans certains problèmes (par exemple, des problèmes de p-médian), le nombre d'activités à localiser est connu. Dans d'autres cas (par exemple, le problème de localisation de l'ensemble couvrant), le nombre d'activités n'est pas connu et est une sortie du problème. Pour des problèmes dans lesquels le nombre d'activités à localiser est connu, nous distinguons entre problèmes de localisation d'une seule activité et de à plusieurs activités. Souvent, les problèmes de localisation d'une seule activité sont beaucoup plus faciles que ceux à plusieurs activités.

2.2.2.3 Capacité limitée ou illimitée des activités

La plupart de modèles de localisation traitent les activités comme ayant une capacité illimitée. Il existe aussi des modèles qui imposent des limites de capacité pour les activités. Dans des modèles, chaque activité a une capacité maximale, ce qui limite le nombre de clients qui peuvent lui affecté. En général, toutes les activités sont identiques et ont la même capacité.

2.2.2.4 Règle d'allocation entre les clients et les activités

L'allocation entre les demandes des clients et les activités est une question cruciale dans la modélisation de problèmes de localisation. Dans beaucoup de cas, les demandes des clients sont affectés à l'activité la plus proche. Mais dans les problèmes avec capacité limitée, il est possible de repartir les demandes d'un client sur plusieurs activités pour que les contraintes de capacité ne soient pas violées. Cela peut entraîner que des demandes sont affectées à d'autres activités que la plus proche. Dans certains modèles on doit gérer

le cas mixte où une partie des demandes est servie par l'activité la plus proche et le reste des demandes est servi par des activités plus loin lorsque l'activité la plus proche est à sa capacité maximale.

2.2.2.5 Types de fonction objectif

Une des caractéristiques importantes dans les problèmes de localisation est l'objectif donné par le décideur. Traditionnellement, nous avons deux types d'objectifs : le premier est la somme des distances pondérées, qui est utilisée pour représenter les coûts de transport. Ce critère, qui est à minimiser, est nommé critère *médian*. Le deuxième type est la distance pondéré maximale entre un client et son activité affectée. Ce critère qui est à minimiser est nommé critère *centre*.

Depuis la fin des années 1970, des chercheurs (par exemple [Erkut et Newman, 1989]) ont également considéré des problème de localisation d'activités désirables ou indésirables. Les activités désirables sont souhaitables d'être situées les plus proches possibles de clients donnés, tandis que les activités indésirables doivent être placées au plus loin.

Un autre type d'objectif assez récent est lié à l'équité. Des modèles avec ce type d'objectif tentent de localiser des activités, de sorte que les distances pondérées entre chaque activité et les clients affectés soient équilibrées. Une autre version de l'équité peut être interprétée comme le fait que chaque activité couvre un même nombre de clients, ou que chaque activité possède une charge équilibrée. Cette version du problème peut être formulé comme le problème de la minimisation de la charge maximale affectée à chaque activité. Le problème dans l'espace euclidien est étudié par [Drezner et Drezner, 2006], et des algorithmes heuristiques sont proposés par [Berman *et al.*, 1989] pour la résolution de ce problème dans un réseau.

2.2.2.6 Problèmes multicritères

La plupart des problèmes de localisation ne mettent en jeu qu'un seul critère. Cependant, de nombreux problèmes de localisation sont multicritères par nature. Un exemple classique est celui de la localisation de déchetterie. Les décharges sont généralement considérées comme des activités indésirables. La plupart des gens ne veulent pas que ces sites soient situés dans leur commune. Par conséquent, il est souhaitable que ces activités soient loin des centres de population. D'autre part, nous tenons à minimiser la distance que les véhicules de transport parcourent entre les sites de collecte de déchets et les déchetteries. Ainsi, résoudre ce problème de localisation revient à trouver un bon compromis entre ces deux critères conflictuels.

L'analyse multicritère de problèmes de localisation a reçu une attention considérable dans un espace réel et dans un réseau ces dernières années. Actuellement, il y a plusieurs problèmes qui sont considérés comme classiques : les problèmes de localisation multicritère des critères de type *médian* dans un espace réel (voir [Hamacher et Nickel, 1996] ; [Puerto et Fernandez, 1999]), et dans un réseau (voir [Hamacher *et al.*, 1998]).

2.2.2.7 Autres facteurs

Il existe aussi d'autres façons pour classifier les problèmes et modèles de localisation : un modèle peut être statique ou dynamique. Dans un modèle statique, des entrées (des demandes, des coûts, des activités existantes...) ne dépendent pas du temps. Dans un modèle dynamique, des entrées peuvent dépendre du temps ; des modèles peuvent aussi être déterministes ou probabilistes ; ou des problèmes peuvent être dans des secteurs privés ou publics... Nous ne détaillerons pas plus dans ce rapport de thèse.

2.2.3 Les problèmes basiques de localisation dans un réseau

Dans tous les problèmes que nous étudions, nous nous concentrons sur des problèmes de localisation dans un réseau. Nous supposons qu'un réseau de n sommets est déjà donné. Des demandes de clients (représentés par des sommets) vont être servies par des activités. Le problème général consiste à localiser de nouvelles activités pour optimiser certains objectifs. Selon ces objectifs, nous avons des modèles différents. Nous étudierons d'abord des problèmes couvrants, puis des modèles qui sont basés sur un objectif de type *centre*, et des modèles basés sur un objectif de type *médian*.

2.2.3.1 Problèmes couvrants

Dans le contexte de problèmes couvrants de localisation, des demandes clients sont généralement affectées aux activités les plus proches pour des raisons économiques. Le service pour un client est évalué par sa distance à l'activité qui lui a été affectée. Si cette distance est inférieure à une distance prédéfinie, le service sera adéquat. Avant de présenter un modèle mathématique de ce problème nous introduisons la notion de couverture. Dans un réseau, chaque sommet de demande i est associé avec un sous-ensemble de sommets N_i (un ensemble de sommets d'activités potentiels j qui peuvent servir (couvrir) le sommet i). Un sommet de demande i est considéré couvert par un site d'activité j si la distance entre le sommet i et le site d'activité j est inférieure ou égale à une distance de couverture. [Schilling *et al.*, 1993] présentent un état de l'art des problèmes couvrants de localisation.

Modèle de localisation de l'ensemble couvrant (the set covering location problem)

Nous commençons par introduire un modèle mathémaitque du problème de localisation de l'ensemble couvrant. L'objectif de ce problème est de minimiser le nombre d'activités à localiser en couvrant toutes les demandes clients. Pour formuler ce problème, nous utilisons les notations suivantes.

Données

$$a_{ij} = \begin{cases} 1 & \text{si le site } j \text{ peut couvrir le sommet } i \\ 0 & \text{sinon} \end{cases} \qquad \forall i \in I, \forall j \in J$$

Variables

$$y_j = \begin{cases} 1 & \text{si une activité est localisée au site } j \\ 0 & \text{sinon} \end{cases} \qquad \forall j \in J$$

Objectif

$$\text{Min} \sum_{j \in J} y_j \tag{2.14}$$

Contraintes

$$\sum_{j \in J} a_{ij} y_j \geq 1, \quad \forall i \in I \tag{2.15}$$

$$y_j \in \{0,1\}, \quad \forall j \in J \tag{2.16}$$

La fonction objectif (2.14) minimise le nombre d'activités à localiser. Les contraintes (2.15) assurent que chaque sommet de demande est couvert par au moins une activité. Les contraintes (2.16) sont des contraintes d'intégralité. Les coefficients a_{ij} peuvent être remplacés par des ensembles N_i comme suit :

- d_{ij}, distance entre le sommet demande i et le site j, $\forall i \in I$, $\forall j \in J$,
- D_c, distance de couverture maximale,
- $N_i = \{j \in J | d_{ij} \leq D_c\}$: ensemble des sites j qui peuvent couvrir le sommet demande i.

Les contraintes (2.15) sont alors remplacées par :

$$\sum_{j \in N_i} y_j \geq 1, \quad \forall i \in I \tag{2.17}$$

Il peut arriver que la fonction objectif (2.14) soit considérée sous une forme plus générale en ajoutant un coût d'installation comme coefficient des variables de décision. Ces deux versions générales du problème de localisation de l'ensemble couvrant sont NP-difficile ([Garey et Johnson, 1979][Megiddo *et al.*, 1983]). En effet, les sous-problèmes pour lesquels $D_c = 1.5$ et $d_{ij} = 1$, pour tous i, j, $\forall i \in I$, $\forall j \in J$, sont équivalents au problème de calcul d'un ensemble dominant dans un graphe.

Il existe un certain nombre de règles pour réduire le nombre de variables y_j du problème (voir [Daskin, 1995] pour une discussion sur ces règles). Par exemple, une variable y_k peut être éliminée du modèle s'il existe une variable y_j telle que $M_k \subset M_j$, où $M_k = \{i | d_{ik} \leq D_c\}$ et $M_j = \{i | d_{ij} \leq D_c\}$. Cette réduction est possible car l'activité j peut couvrir tous les sommets demande que l'activité k peut couvrir. Donc, nous pouvons dire que l'activité j domine l'activité k. Une contrainte, correspondant à un sommet demande h, dans l'ensemble de contraintes (2.15) peut être éliminée s'il existe un sommet demande i, tel que $N_i \subset N_h$. Cette réduction de contraintes est possible parce que la contrainte (2.15) liée au sommet h est redondante. C'est-à-dire que si le sommet i est couvert, le sommet h le sera également. Une autre réduction de contraintes est possible s'il n'existe qu'une activité pour couvrir certains sommets demande. Par exemple, pour sommet i, si $\sum_j a_{ij} = 1$, nous pouvons fixer $y_{j'} = 1$ tel que $a_{ij'} = 1$. Et nous pouvons éliminer toutes

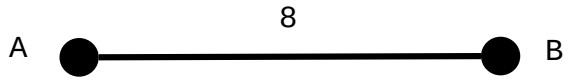

FIGURE 2.1 – Exemple de suboptimalité

les contraintes dans lesquelles $y_{j'}$ apparaît. Ces règles de prétraitement visant à réduire la taille des problèmes à résoudre peuvent être utilisées en amont d'une résolution exacte ou d'une résolution heuristique.

La formulation (2.14)-(2.16) suppose que les sites candidats ne sont localisés que sur les sommets de réseau. Les solutions issues de cette formulation ne garantissent pas "l'optimalité". Puisque nous pouvons obtenir une solution meilleure si nous autorisons la localisation d'activités sur des arcs du réseau. Prenons l'exemple de la Figure 2.1 : nous avons deux sommets demande A et B, la distance entre eux est 8 et la distance de couverture est 5. Si nous limitons la localisation de sites d'activités aux sommets A et B, alors nous avons besoin de deux activités. Toutefois, si nous autorisons de localiser des activités partout dans ce réseau, alors nous n'avons besoin qu'une activité placée entre A et B. [Church et Meadows, 1979] prouvent que si nous ajoutons un ensemble fini de points particuliers (points d'intersection de réseau) dans l'ensemble des sommets, la solution correspondante a le même nombre d'activités que si des activités sont autorisées à être placées partout.

Concernant les méthodes pour résoudre ce problème, [Caprara *et al.*, 2000] présente un revue sur les algorithme heuristiques et exactes en décrivant leur principales caractéristiques et en accompagnant par une comparaison expérimentale. Parmi les heuristiques présentées, selon leur tests sur les instances de la bibliothèque de Beasley (Beasley's OR Library), l'heuristique proposée par [Lorena et Lopes, 1994] est la plus rapide en temps de calcul, et l'heuristique proposée par [Caprara *et al.*, 1999] est celle qui donne les meilleurs résultats. Les approches exactes les plus efficaces pour ce problème sont les procédures par séparation et évaluation (Branch and Bound) dans lesquels les bornes inférieures sont calculées en résolvant la relaxation de la programmation linéaire du problème. Ils comparent les meilleures algorithmes exactes [Beasley, 1987][Beasley et Jörnsten, 1992][Balas et Carrera, 1996] avec les solveurs CPLEX 4.0.8 et MINTO 2.3, et ils trouvent que en générales ces solveurs peuvent trouver la solution optimale plus vite que les autres méthodes exactes.

Modèle de localisation de l'ensemble couvrant maximum (the maximum covering location problem)

Dans le problème de localisation de l'ensemble couvrant, l'objectif est de couvrir tous les sommets demande avec un coût minimal, et sans contrainte de budget. En plus, dans ce modèle, les sommets demande sont traités avec de la même façon : il n'y a pas de poids sur les sommets. Néanmoins, il existe de nombreuses situations pratiques dans la planification d'activités, un budget existe. Par exemple, de nombreux districts voudraient voir installée une école primaire où tous les élèves de ce district pourraient y arriver en ne dépassant pas une certaine distance de marche. Toutefois, satisfaire une telle exigence peut nécessiter plus d'écoles que le district n'est prêt à construire. Le problème de loca-

lisation de l'ensemble couvrant maximum, formulé par [Church et ReVelle, 1974], vise à résoudre le problème de localisation avec une limite sur le nombre d'activités (contrainte de budget). Dans ce problème, l'objectif est de maximiser des demandes de clients qui sont couvertes avec un nombre prédéfini d'activités. Ce problème suppose qu'il n'y a peut-être pas suffisamment d'activités pour couvrir tous les sommets demande, et nous cherchons des localisations d'activités qui couvrent le plus grand nombre possible de demandes. Pour formuler ce problème, nous utilisons les notations suivantes.

Données

$$a_{ij} = \begin{cases} 1 & \text{si le site } j \text{ peut couvrir le sommet } i \\ 0 & \text{sinon} \end{cases} \quad ,\forall i \in I, \forall j \in J$$

$h_i =$ la demande du sommet i $\quad ,\forall i \in I,$

$p =$ le nombre d'activités à localiser.

Variables

$$y_j = \begin{cases} 1 & \text{si l'activité est localisée au site } j \\ 0 & \text{sinon} \end{cases} \quad ,\forall j \in J$$

$$x_i = \begin{cases} 1 & \text{si le sommet i est couvert} \\ 0 & \text{sinon} \end{cases} \quad ,\forall i \in I$$

Objectif

$$\text{Max} \sum_{i \in I} h_i x_i \tag{2.18}$$

Contraintes

$$x_i \leq \sum_{j \in J} a_{ij} y_j, \quad \forall i \in I \tag{2.19}$$

$$\sum_{j \in J} y_j = p, \tag{2.20}$$

$$y_j \in \{0,1\}, \quad \forall j \in J \tag{2.21}$$

$$x_i \in \{0,1\}, \quad \forall i \in I \tag{2.22}$$

La fonction objectif (2.18) maximise les demandes clients couvertes par les activités. Les contraintes (2.19) expliquent qu'un sommet demande i est considérée couvert s'il existe au moins une activité ouverte qui peut le couvrir. La partie droite de cette contrainte est identique avec la partie gauche de contrainte (2.15) : elle donne le nombre d'activités qui peuvent couvrir le sommet i. Les contraintes (2.20) impliquent que le nombre d'activités est égale à p. Les contraintes (2.21) et (2.22) sont les contraintes d'intégralité des variables de décision.

Le problème de localisation de l'ensemble couvrant maximum est NP-difficile ([Megiddo *et al.*, 1983]). La première heuristique à avoir été proposée est appelée *Ajout Glouton* (Greedy Adding, [Church, 1974]). Cette heuristique commence avec une solution vide et ajoute une activité

dans cette solution à chaque étape. Concrètement, cet algorithme choisit une première activité qui couvre la plus grande quantité de demandes. Ensuite à chaque étape, cet algorithme choisit une activité parmi celles qui ne sont pas encore couvertes et qui peut couvrir la plus grande quantité de demandes sur des sommets demande restants (qui ne sont pas couverts par des activités déjà installées). Le procédure se termine quand p activités sont sélectionnées ou toutes les demandes sont couvertes.

La deuxième heuristique est basée sur la première heuristique : elle est appelée *Ajout Glouton avec Substitution* (Greedy Adding with Substitution, [Church, 1974]). Cet algorithme détermine aussi une nouvelle localisation d'activité à chaque étape, mais il cherche à améliorer la solution courante à chaque itération en remplaçant une activité déjà affectée par une autre activité libre (un sommet non encore couvert). S'il existe des améliorations, la permutation choisie est celle qui améliore le plus la fonction objectif.

Ces deux heuristiques présentées ne garantissent pas que la solution trouvée soit optimale. Nous ne savons même pas si la solution trouvée est loin de solution optimale ou pas. En utilisant la relaxation lagrangienne, nous pouvons obtenir une borne supérieure sur la valeur de la fonction objectif. [Galvao et ReVelle, 1996] et [Galvao *et al.*, 2000] utilisent la relaxation lagrangienne pour résoudre ce problème. Concernant les méthodes exactes, [Dwyer et Evans, 1981] développent une procédure par séparation et évaluation pour le cas particulier de ce problème où tous les demandes des clients où ont la même poids et [Downs et Camm, 1996] proposent un algorithme exact en utilisant la résolution basée sur le problème dual.

2.2.3.2 Problème p-centre

Le problème p-centre ([Hakimi, 1964][Hakimi, 1965]) consiste à déterminer la localisation de p activités dans un réseau tel que la distance pondérée maximale de n'importe quelle sommet demande à une activité la plus proche soit minimale. Nous distinguons les problèmes dans lesquels les activités peuvent être localisées partout dans le réseau, et les problèmes dans lesquels les activités ne peuvent être localisées que sur des sommets dans le réseau. Ces premiers problèmes sont appélés problèmes de type p-centre *absolu*, tandis que ces seconds problèmes sont appelés problèmes de type p-centre *sommet*. Ces deux versions peuvent avoir des poids sur des sommet demande. Ici, nous ne nous intéressons qu'aux problèmes p-centre *sommet*. Une formulation en programmation linéaire en nombres entières du problème est la suivante.

Données

h_i = demande au sommet i,
d_{ij} = distance entre le sommet i et le sommet j,
p = le nombre d'activités à localiser.

Variables

$x_{ij} = \begin{cases} 1 & \text{si le sommet } i \text{ est affecté à l'activité } j \\ 0 & \text{sinon} \end{cases}$

$y_j = \begin{cases} 1 & \text{si l'activité } j \text{ est ouverte} \\ 0 & \text{sinon} \end{cases}$

$W =$ distance pondérée maximale entre le sommet i et l'activité la plus proche.

Objectif

$$\text{Min } W \tag{2.23}$$

Contraintes

$$W \geq \sum_{j \in J} h_i d_{ij} x_{ij}, \forall i \in I \tag{2.24}$$

$$\sum_{j \in J} x_{ij} = 1, \quad \forall i \in I \tag{2.25}$$

$$x_{ij} \leq y_j, \quad \forall i \in I, j \in J \tag{2.26}$$

$$\sum_{j \in J} y_j = p, \tag{2.27}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, \forall j \tag{2.28}$$

$$y_j \in \{0, 1\}, \quad \forall j \tag{2.29}$$

Les contraintes (2.24) définissent la fonction objectif à minimiser. Les contraintes (2.25) assurent que chaque sommet demande soit affecté à une activité. Les contraintes (2.26) déclarent que tout sommet i peut être affecté à une activité j seulement si l'activité j est ouverte. Les contraintes (2.27) imposent le nombre d'activités à localiser. Enfin, les contraintes (2.28) et (2.29) sont des contraintes d'intégralité des variables de décision.

Quand la valeur de p est fixée, le problème de p-centre *sommet* peut être résolu en temps $O(n^p)$ avec n est le nombre de sommets dans le réseau, puisque nous pouvons énumérer tous les ensembles possibles de sites d'activité dans ce temps. Mais même si les valeurs de n et p ne sont pas grandes, cette énumération n'est pas réaliste. Quand la valeur de p est variable, le problème de p-centre *sommet* est NP-complet ([Kariv et Hakimi, 1979a]).

Une manière de résoudre le problème p-centre *sommet* consiste à résoudre une série de problèmes de recherche d'un ensemble couvrant ([Minieka, 1970]) : itérativement choisir un seuil pour le rayon (la valeur de la solution pour le problème p-centre) et vérifier si tous les sommets demande peuvent être couverts à l'intérieur de cette distance en n'utilisant pas plus de p activités. Si c'est le cas, on diminue ce seuil à l'itération suivante, sinon on l'augmente. Cependant, cette méthode n'est pas efficace et n'est capable de résoudre des instances de grande taille en pratique.

Un certain nombre d'heuristiques classiques proposées dans la littérature exploitent la relation existante entre le problème p-centre et un autre problème appelé le problème de recherche d'ensemble dominant ([Hochbaum et Shmoys, 1985][Martinich, 1988]). Étant donné un graphe complet $G = (V, E)$, un ensemble dominant S de G est un sous-ensemble

de V tel que chaque sommet dans $V \backslash S$ est adjacent dans G à un sommet de S. Le problème est de trouver un ensemble dominant S de cardinalité minimale. Pour une solution $X = \{v_1, ..., v_p\}$ du problème p-centre, il existe évidemment une arête (v_a, v_c) telle que $d(v_a, v_c) = f(x)$, où v_a est un sommet d'activité et v_c est un sommet demande, $d(v_a, v_c)$ est leur distance et $f(x)$ est la valeur optimale de la fonction objectif. Nous pouvons supprimer tous les liens du problème initial dont les distances sont plus grandes que $f(x)$. Alors X est un ensemble dominant minimal dans un tel graphe. Si X est une solution optimale, alors le graphe correspondant avec toutes les arêtes dont les longueurs sont inférieures ou égales à $f(x)$ est appelé un graphe de goulot d'étranglement. Par conséquent, afin d'exploiter cette relation entre les deux problèmes, il faut chercher le graphe de goulot d'étranglement et trouver son ensemble dominant minimal. Dans les deux heuristiques ([Hochbaum et Shmoys, 1985][Martinich, 1988]), toutes les distances sont triées d'abord, puis les graphes G_t contenant les t arêtes les plus courtes sont construits, et les cardinalités des ensembles dominants dans ces graphes sont trouvés approximativement. Les deux heuristiques s'arrêtent quand le cardinalité approché de l'ensemble dominant est égale à p. Elles diffèrent par l'ordre dans lequel les sous-problèmes sont résolus. Mais malheureusement, le problème est lui-même NP-difficile et, dans ce sens, cette réduction n'est pas très utile.

Trois heuristiques classiques nommées Glouton, Alternative et Substitution, qui sont souvent utilisés pour résoudre le problème de p-médian, peuvent être adaptées simplement pour résoudre le problème p-centre. Avec l'algorithme Glouton, une première activité est localisée dans le but de minimiser le coût maximum, c'est-à-dire, un problème 1-centre est d'abord résolu. Les activités sont ensuite ajoutées une par une jusqu'à ce que p soient placées : chaque fois l'activité qui réduit le plus le coût maximum est sélectionnée. [Dyer et Frieze, 1985] proposent une variante de Glouton où le premier centre est choisi aléatoirement. Avec l'algorithme Alternative, la première itération consiste à localiser p activités choisis dans V, puis à affecter des demandes à l'activité la plus proche et résoudre le problème de 1-centre pour ensemble des demandes de chaque activité. Ensuite, la procédure est itérée avec de nouveaux sites d'activités jusqu'à ce qu'il n'y ait plus de changements dans des affectations. Cette heuristique consiste à alternativement localiser des activités puis leur affecter des demandes. Dans l'algorithme Substitution, une solution à p activités est donnée initialement, puis, les activités sont remplacée itérativement, une par une, par des sommets demandes dans le but de réduire de coût maximum. Cette procédure de recherche locale s'arrête quand il n'existe plus de changement d'activité qui diminue la valeur de la fonction objectif. Dans les versions de type "multi-démarrage" de Substitution, la procédure est répétée un nombre de fois donné et la meilleure solution est gardée. Une implémentation efficace est proposée par [Mladenović *et al.*, 2003].

Des heuristiques de type recherche à voisinage variable et recherche tabou sont aussi présentées par [Mladenović *et al.*, 2003] pour le problème de p-centre. Ces deux méthodes utilisent la structure de voisinage de substitution de sommets.

Concernant les méthodes exacte, [Minieka, 1970] propose un algorithme qui s'appuie en résolvant une séquence finie des problèmes de l'ensemble couvrant. Son idée est de choisir un seuil de distance (la distance couvrante) comme un rayon et de vérifier si tous les points de demande de clients sont couverts dans ce rayon en utilisant pas plus que les p activités. Basé sur l'idée de Minieka, [Daskin, 1995] développe un algorithme pour résoudre ce pro-

blème optimalement en utilisant la méthode de dichotomie qui réduit systématiquement l'écart entre les bornes supérieure et inférieure de la solution optimale, et donc trouve la distance couvrante optimale. [Ilhan et Pinar, 2001] propose une méthode exacte qui se compose de deux phases, et pour la première fois, la taille de l'instance qui peut être résolu optimalement peut monter jusqu'à 900 sommets de demande.

2.2.3.3 Problème p-médian

Le problème p-médian ([Hakimi, 1964][Hakimi, 1965]) consiste à localiser p activités dans un réseau tel que le coût total est minimal. Le coût de service pour un sommet i est lié à la demande du sommet i et à la distance entre le sommet i et le site d'activité le plus proche.

[Hakimi, 1965] prouve que si p activités sont localisables seulement aux sommets du réseau, alors il existe au moins une solution optimale qui a la même qualité que la solution optimale du problème où les activités sont autorisées à être localisées partout dans le réseau. Grâce à cette propriété, nous pouvons nous concentrer sur la recherche d'ensembles de p sommets dans un graphe. Le problème de p-médian peut être défini comme suit : étant donné un graphe complet ou un réseau $G = (V, E)$, nous cherchons à localiser p activités sur des sommets du réseau, noté comme V_p, telle que la somme des distances totales entre chaque sommet dans V/V_p et son plus proche sommet dans V_p est minimisée.

Les modèles de p-médian pondéré et non-pondéré diffèrent par la prise en compte ou non des demandes des sommets. Le problème pondéré peut être formulé en utilisant les notions suivantes :

Données

$h_i =$ la demande du sommet i,
$d_{ij} =$ la distance entre le sommet i et le sommet j,
$p =$ le nombre d'activités à localiser.

Variables

$$x_{ij} = \begin{cases} 1 & \text{si le sommet } i \text{ est affecté à l'activité } j, \\ 0 & \text{sinon.} \end{cases}$$
$$y_j = \begin{cases} 1 & \text{si l'activité } j \text{ est ouverte,} \\ 0 & \text{sinon.} \end{cases}$$

Objectif

$$\text{Min} \sum_{i \in I} \sum_{j \in J} h_i d_{ij} x_{ij} \tag{2.30}$$

Contraintes

$$\sum_{j \in J} x_{ij} = 1, \quad \forall i \in I \tag{2.31}$$

$$x_{ij} \leq y_j, \quad \forall i \in I, \forall j \in J \tag{2.32}$$

$$\sum_{j \in J} y_j = p, \tag{2.33}$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in I, \forall j \in J \tag{2.34}$$

$$y_j \in \{0,1\}, \quad \forall j \in J \tag{2.35}$$

La fonction objectif (2.30) minimise la distance pondérée totale entre chaque sommet demande et l'activité ouverte la plus proche. Les contraintes (2.31) assurent que chaque sommet demande soit affecté à une activité. Les contraintes (2.32) déclarent que le sommet i peut être affecté à l'activité j seulement si celle-ci est ouverte. Les contraintes (2.33) imposent le nombre d'activités à localiser. Les contraintes (2.34) et (2.35) sont des contraintes d'intégralité des variables de décision.

Il est intéressant de noter que les contraintes (2.32) sont des contraintes fortes qui relient les variables de localisation et les variables d'affectation. Elles peuvent être remplacées par des versions plus faibles :

$$\sum_{i \in I} x_{ij} - n y_j \leq 0, \quad \forall j \in J. \tag{2.36}$$

où n est le nombre de sommets du demande ($n = |I|$). Si l'activité j n'est pas ouverte, alors toutes les variables d'affectation utilisant cette activité doivent être 0. Cette formulation regroupe des contraintes (2.32) et réduit le nombre de contraintes. Cependant, il est bien connu que la borne inférieure obtenue par la relaxation continue de cette version (on remplace les contraintes d'intégralité (2.34) et (2.35) par des contraintes de non-intégralité) est moins efficace que celle associée à la version avec les contraintes (2.32).

Comme pour le problème p-centre *sommet*, quand la valeur de p est fixe, le problème p-médian peut être résolu en temps $O(N^p)$ puisque nous pouvons énumérer tous les ensembles possibles de sites d'activité dans ce temps. Quand la valeur de p est variable, le problème de p-médian est NP-complet ([Kariv et Hakimi, 1979b]).

Initialement, trois heuristiques ont été proposées dans la littérature : Glouton ([Kuehn et Hamburger, 196
Alternative [Maranzana, 1964]) et Substitution ([Teitz et Bart, 1968]). Dans l'algorithme Glouton ([Daskin, 1995]), nous plaçons la première activité, puis la deuxième activité,..., et la p-ième. À la k-ième ($1 \leq k \leq p$) itération, avec les $k-1$ activités déjà localisées dans les itérations précédentes, nous cherchons une activité j^k dans l'ensemble des activités pas encore localisées, et telle que son intégration minimise la fonction objectif. L'algorithme Alternative ([Maranzana, 1964]) commence avec un ensemble de p sommets générés aléatoirement comme activités : $V_p = \{m_j : 1 \leq j \leq p\}$ avec m_j le $j$ème activité. Puis l'algorithme partitionne les sommets restants $V \backslash V_p$ en les affectant aux activités les plus proches. Pour chaque partition, l'algorithme détermine un centre de gravité c_j. Si pour toutes les partitions j, $m_j = c_j$, alors l'algorithme se termine, et V_p est le p-médian. Sinon, nous mettons $m_j = c_j$, et répétons le processus : de nouvelles partitions sont alors recréées. Cet algorithme ne converge pas toujours vers un optimum et la qualité de la solution dépend du choix initial des p-médian. L'heuristique de Substitution ([Teitz et Bart, 1968]) est proposée en utilisant la substitution de sommets pour trouver le p-médian. Cette méthode commence par une sélection d'un ensemble de sommets initiaux $V_p = \{m_j : 1 \leq j \leq p\}$. Pour chaque sommet $v_i \notin V_p$, l'algorithme trouve l'activité affecté $m_j \in V_p$, et calcule le degré d'amélioration si m_j est remplacé par v_i. L'algorithme choisit l'amélioration la plus importante et réalise la subsitution associée pour obtenir un nouveau p-médian. Le

processus se répète sur le nouveau p-médian. Lorsqu'il n'y a plus d'amélioration à faire, l'algorithme se termine.

La relaxation lagrangienne est une méthode de résolution qui consiste à relaxer des contraintes et à introduire des multiplicateurs lagrangiens. Elle permet de transformer un problème plus restrictif en un problème moins restrictif. En utilisant la relaxation lagrangienne, nous devons être capable de résoudre un problème relâché facilement pour des valeurs fixes de multiplicateurs lagrangiennes, trouver des solutions primales et réalisables à partir de solution relâchée, et trouver des nouveaux et meilleurs multiplicateurs lagrangiens. Pour résoudre le problème de p-médian à l'aide de la relaxation lagrangienne, nous pouvons relâcher les contraintes (2.31) ou les contraintes (2.32). Ces deux relaxations sont comparées par [Christofides et Beasley, 1982] qui trouvent que la première relaxation est supérieure.

L'avantage de la relaxation lagrangienne est qu'elle propose une borne inférieure sur la solution optimale. Cette borne peut être utilisée dans une méthode exacte comme, par exemple, une procédure par séparation et évaluation. Pour le problème p-médian, il existe notamment des procédures de séparation et évaluation telles que celle de [Järvinen *et al.*, 1972].

Le problème p-médian peut être généralisé en ajoutant une capacité Q_j pour un sommet candidat de l'activité j, et en imposant que la somme des demandes affectées à une activité ne peut pas dépasser sa capacité. Cette généralisation est connue sous le nom du problème p-médian avec capacité limitée ([Hansen et Jaumard, 1997]). En raison des contraintes de capacité, les algorithmes développés pour le problème p-médian ne peuvent plus être adaptés au problème p-médian avec capacité limitée.

Le problème p-médian avec capacité limitée peut être formulé comme le problème p-médian en remplaçant les contraintes (2.32) par :

$$\sum_{i \in I} h_i x_{ij} \leq Q_j y_j, \quad \forall j \in J \tag{2.37}$$

Les contraintes (2.37) ont deux effets. Elles imposent que la somme des demandes affectées à l'activité j ne doit pas excéder sa capacité et elles interdisent l'affectation de demandes à des activités inactives (pour lesquelles $y_j = 0$).

Nous pouvons remarquer que, en utilisant cette formulation, une activité peut être placée en dehors du groupe des sommets demande qui lui est affecté (*i.e.* $y_j = 1$ et $x_{jj} = 0$). Pour éviter cet effet, certains auteurs (par exemple, [Baldacci *et al.*, 2002]) supposent que $d_{jj} = 0$, $\forall j \in J$, et imposent que $x_{jj} = 1$ si et seulement si sommet j est médian. Cette hypothèse peut changer la solution optimale. Considérons un exemple dans la Figure 2.2, avec le nombre de demande $n = 3$ et le nombre d'activités $p = 2$. Pour satisfaire toutes les demandes, les deux activités doivent être installées aux sommets 2 et 3. Toutefois, pour obtenir une solution faisable, le seul moyen est d'affecter le sommet 1 à l'activité située au sommet 2, et d'affecter le sommet 2 à l'activité située au sommet 3. Si nous imposons $x_{jj} = y_j$, il y aura pas de solution faisable dans cet exemple.

Il y a peu de méthodes pour résoudre ce type de problème avec capacité. Un algorithme pour reformuler ce problème comme un problème d'affectation est proposé par [Ross et Soland, 1977]. Des heuristiques capables de résoudre des problèmes avec 100 sommets demande et 20 activités en quelques dizaines de secondes sont présentées par [Mulvey et Beck, 1984].

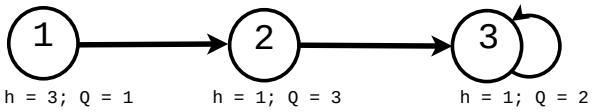

FIGURE 2.2 – Un exemple avec $N = 3$ et $p = 2$

[Maniezzo *et al.*, 1998] adoptent la programmation évolutive, [Golden et Skiscim, 1986] utilisent le recuit simulé, tandis que [Osman et Christofides, 1994] exploitent le recuit simulé et la recherche tabou pour résoudre des problèmes avec 100 sommets demande et 10 activité en moins d'une heure. [Ceselli, 2003] présente deux algorithmes exacts, une méthode par séparation et évaluation (branch-and-bound) et une méthode par séparation et évaluation avec une génération de colonnes à chaque nœud de l'arbre (branch-and-price).

2.2.4 Problème de localisation avec charge équilibrée

Le problème de localisation avec charge équilibrée (PLCE) ressemble au problème de sectorisation que nous abordons dans cette thèse. Dans le PLCE, nous cherchons à implanter p activités sur des sommets du réseau (un graphe complet) de telle sorte que chaque activité ait une charge équilibrée. Chaque sommet demande est affecté à l'activité la plus proche. Dans la littérature, il y a peu de références qui traitent ce problème. [Berman *et al.*, 2009] considèrent une solution équitable du point de vue de l'activité : la charge attribuée à chaque activité doit être équilibrée. Concrètement, ils considèrent le problème de localisation de p activités pour servir un ensemble de sommets demande, de telle sorte que la demande maximale attribuée à chaque activité soit minimale. [Kalcsics *et al.*, 2002] tiennent compte de plusieurs problèmes de localisation ordonnés. [Berman *et al.*, 2009] en proposent la formulation suivante :

Données

$h_i =$ la demande du sommet i,
$d_{ij} =$ la distance entre le sommet i et le sommet j,
$p =$ le nombre d'activités à localiser.

Variables

$$x_{ij} = \begin{cases} 1 & \text{si le sommet } i \text{ est affecté à l'activité } j, \\ 0 & \text{sinon.} \end{cases}$$
$$y_j = \begin{cases} 1 & \text{si l'activité } j \text{ est ouverte,} \\ 0 & \text{sinon.} \end{cases}$$

Objectif

$$\text{Min } z \tag{2.38}$$

Contraintes

$$\sum_{i \in I} h_i x_{ij} \leq z, \quad \forall j \in J \tag{2.39}$$

$$\sum_{j \in J} x_{ij} = 1, \quad \forall i \in I \tag{2.40}$$

$$x_{ij} \leq y_j, \quad \forall i \in I, \forall j \in J \tag{2.41}$$

$$\sum_{j \in J} y_j = p, \tag{2.42}$$

$$\sum_{k=1}^{n} d_{ik} x_{ik} + (F - d_{ij}) y_j \leq F, \quad \forall i \in I, \forall j \in J \tag{2.43}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in I, \forall j \in J \tag{2.44}$$

$$y_j \in \{0, 1\}, \quad \forall j \in J \tag{2.45}$$

où F est un très grand nombre ($F \geq max_{ij}\{d_{ij}\}$).

La fonction objectif (2.38) et les contraintes (2.39) minimisent la charge maximale des activités. Les contraintes (2.40) assurent que chaque sommet demande soit affecté à une activité. Les contraintes (2.41) impliquent que chaque sommet i soit affecté à l'activité j seulement si celle-ci est ouverte. Les contraintes (2.42) imposent le nombre d'activités à localiser. Nous allons montrer que les contraintes (2.43) garantissent que chaque sommet demande est affecté à une activité la plus proche.

Soit $J = \{j | y_j = 1\}$. Pour $j \notin J$, (2.43) est toujours vrai parce que $y_j = 0$. Pour $j \in J$, par (2.41), $x_{ik} = 0$ pour $k \notin J$, par conséquent, la somme de la partie gauche de (2.43) peut être écrit comme $\sum_{k \in J} d_{ik} x_{ik}$, et (2.43) peut être écrite comme

$$\sum_{k \in J} d_{ik} x_{ik} \leq d_{ij} \tag{2.46}$$

Si $x_{ik} = 1$ pour $d_{ik} > min_t\{d_{it}\}$, la contrainte (2.46) sera violée pour $d_{ij} = min_t\{d_{it}\}$. Donc, x_{ik} peut être égal à 1 seulement quand d_{ik} est la distance minimale. La fonction objectif (2.38) et les contraintes (2.39) traitent le cas quand une demande a plusieurs activités ayant la même distance : il est alors préférable d'affecter cette demande à l'activité ayant une charge la plus faible. Les contraintes (2.44) et (2.45) sont les contraintes d'intégralité des variables de décision.

Si toutes les distances entre paires de sommets sont distinctes, alors pour chaque sous-ensemble S de p sommets il y a seulement une affectation possible. Par conséquent, en tenant compte de $O(n^p/p!)$ sous-ensembles de p sommets, PLCE peut être résolu par énumération complète en temps $O(n^{p+1}/(p-1)!)$ sur un réseau général. En particulier, dans ce cas, PLCE est polynomialement résoluble pour toute valeur fixe de p.

Si les distances entre paires de sommets ne sont pas distinctes, alors pour chaque sous-ensemble S, nous devons appliquer un algorithme pour calculer sa charge maximale. Le dernier problème est un cas particulier du problème d'ordonnancement du makespan minimum de n tâches sur p machines indépendantes et parallèles ([Horowitz et Sahni, 1976]).

Plus précisément, chaque sommet demande est considéré comme une tâche, et chaque activité dans S est considérée comme une machine. Le temps de traitement de la tâche i sur la machine j est w_i pour $j \in J$, et ∞ sinon. Un algorithme exact pseudo-polynomial en $O(\min[nW, p^n])$ avec $W = \sum_{i=1}^{n} w_i$, a été présenté par [Horowitz et Sahni, 1976]. Donc, en considérant $O(n^p/p!)$ sous-ensembles de p sommets, PLCE peut être résolu par énumération complète en $O(n^{p+1}W/(p-1)!)$ sur un réseau général. [Berman *et al.*, 2009] montrent que PLCE est NP-difficile au sens faible lorsque $p = 2$ et G est un arbre d'étoiles (star tree). Considérons un arbre d'étoiles avec un sommet de centre v_0 et n sommets feuille $v_1, ..., v_n$. Chaque sommet feuille est connecté à v_0 avec une arête de longueur 1 et nous mettons $w_0 = 0$. Nous pouvons voir que, lorsque $p = 2$, déterminer si la valeur de la solution optimale est bornée par $W/2 = \sum_{i=1}^{n} w_i/2$, est équivalent au problème de partition ([Garey et Johnson, 1979]).

[Berman *et al.*, 2009] appliquent une recherche locale, une recherche locale améliorée, et une recherche tabu pour résoudre ce problème. La recherche locale est construite selon le principe proposé par [Teitz et Bart, 1968] pour résoudre le problème de p-médian. La recherche locale n'est pas très efficace, parce que quand il y a plusieurs activités qui possèdent une même charge maximale, l'algorithme s'arrête. Donc, dans la recherche locale améliorée, un classement est établi entre deux solutions qui ont une même charge maximale. La recherche tabou ([Glover et Laguna, 1997]) procède à partir d'une solution de recherche locale améliorée. Ces trois algorithmes sont expérimentés et comparés. En termes de qualité de solution, la solution trouvée par la recherche locale est en moyenne 20% moins bonne que la meilleur solution connue, tandis que pour la recherche locale améliorée ce pourcentage est de 0.8%, et pour la recherche tabou 0, 16%. Toutefois, en termes de temps d'exécution, la recherche locale améliorée est environ 5 fois plus lente que la recherche locale. La recherche tabou est environ 3.5 fois plus lente, que la recherche locale améliorée.

2.2.5 Liens avec le problème de sectorisation

Après les études sur les différents problèmes de localisation, nous remarquons de fortes similitudes avec le problème de sectorisation et nous pouvons appliquer les modèles des problèmes de localisation au problème de sectorisation : des éléments basiques dans le problème de sectorisation peuvent être considérés comme des points de demandes dans les problèmes de localisation-allocation ; des pôles à définir dans le problème de sectorisation peuvent être considérés comme des sites d'activité à localiser ; les informations statistiques de chaque élément basique peuvent être considérés comme des poids des points de demandes. Ces deux problème n'ont pas de même objectif et dans les problèmes de localisation, la contrainte de connexité n'est pas prise en compte. Dans la Section 3.2.1, les différents modèles de localisation qui ressemblent proche du problème de sectorisation (le p-médian et le problème de localisation avec charge équilibre) serons appliqués au problème de sectorisation.

2.3 Problèmes de partitionnement de graphes

Les problèmes de partitionnement de graphes apparaissent dans un grand nombre de problèmes d'ingénierie : répartition de charge dans des machines parallèles, conception de circuits intégrés électronique, segmentation d'image, exploration de données, optimisation du trafic aérien, ...

Ces problèmes peuvent être formulés de la façon suivante. Soit un graphe $G = (V, E)$, où V est l'ensemble des sommets et E est l'ensemble des arrêtes qui relient des paires de sommets. Les sommets et les arrêtes peuvent être pondérés, où $poids(v)$ est le poids du sommet v et $poids(u, v)$ est le poids de l'arrête (u, v). Le problème du partitionnement de graphes consiste à diviser G en k partitionnement disjoints. On peut partitionner les sommets ou bien les arrêtes. Dans la plupart des applications, on ne s'intéresse qu'au partitionnement des sommets du graphe.

Soient un graphe $G = (V, E)$ et un ensemble de k sous-ensembles de V, notés $P_k = \{V_1, ..., V_i, ..., V_k\}$. On dit que P_k est une partition de G si :

- aucun sous-ensemble $V_i \subseteq V$ appartenant à P_k n'est vide ;
- les sous-ensembles de V qui sont élément de P_k sont disjoints deux à deux ;
- l'union de tous les éléments de P_k est égale à V, i.e.

$$V = \bigcup_{j=1}^{k} P_j.$$

Les éléments V_i de P_k sont appelés les parties de la partition. Le nombre k est appelé le cardinal de la partition, ou encore le nombre de parties de la partition. Dans le cas $k = 2$, on a le problème de bissection.

Pour mesurer la taille des parties d'une partition, la notion de "balance" de partitionnement est introduite. Soient un graphe $G = (V, E)$ et une partition $P_k = \{V_1, V_2, ..., V_k\}$ de ce graphe en k parties. Le poids moyen d'une partie V_i de P_k est

$$poids_{moy} = \frac{\sum_{v \in V} poids(v)}{k}.$$

La balance $bal(P_k)$ de la partition P_k est égale au ratio entre le poids maximum de V_i et le poids moyen :

$$bal(P_k) = \frac{\max_{1 \leq i \leq k} \sum_{v \in V_i} poids(v)}{poids_{moy}}.$$

Pour avoir un partitionnement équilibré, on impose généralement une contrainte sur la valeur de la balance pour autoriser qu'une petite valeur, typiquement $bal(P_k) \in [1; 1,05]$.

Intéressons nous maintenant à la fonction objectif souvent considéré pour les problèmes de partitionnement de graphes.

Soient deux sous-ensembles $V_a \in V$ et $V_b \in V$. On définit le coût d'une coupe entre ces deux sous-ensembles par :

$$coupe(V_a, V_b) = \sum_{u \in V_a, v \in V_b} poids(u, v).$$

La fonction objectif la plus simple et la plus souvent utilisée en partitionnement de graphes est le coût total de coupe d'une partition. Cette fonction, à minimiser, est la somme des poids des arêtes entre les parties de la partition P_k :

$$coupe(P_k) = \sum_{i<j} coupe(V_i, V_j).$$

Le problème classique du partitionnement d'un graphe consiste alors à trouver une partition en k parties qui minimise la fonction de coût de coupe et satisfasse les contraintes sur la balance de la partition. Les problèmes de partitionnement de graphes sont bien souvent NP-difficile ([Garey *et al.*, 1976]).

Il y a deux catégories générales de méthodes pour le problème de partitionnement de graphes : des méthodes exactes qui calculent la partition optimale, et des méthodes heuristiques qui essayer de calculer rapidement une solution approchée. Dans la suite, quelque méthodes heuristiques classiques sont brièvement présentées : les méthodes d'expansion de région, la méthode spectrale, la méthode multi-niveaux et l'algorithme d'affinage de Kernighan-Lin.

Les méthodes d'expansion de région (graph growing) sont introduites par [Karypis et Kumar, 1998]. Ces méthodes de bissection, consistent à créer itérativement un ensemble E rassemblant la moitié des sommets du graphe en terme de poids. Cet ensemble E est initialisé en choisissant aléatoirement un sommet dans le graphe. Pendant l'exécution de l'algorithme, les sommets du graphe sont répartis en trois ensembles : l'ensemble E, l'ensemble des sommets adjacents à E, noté F, et l'ensemble des sommets restants, noté R. À chaque itération, les sommets adjacents aux sommets de l'ensemble E sont ajoutés à E. Le processus s'arrête lorsque E contient un ensemble de sommets qui représente la moitié du poids total des sommets du graphe. Cette méthode est très simple à mettre en œuvre, et très rapide en temps d'exécution, la qualité de la partition obtenue dépend principalement du sommet de départ.

La méthode spectrale a été très utilisée pour résoudre le problème de partitionnement de graphes avant l'arrivée des méthodes multi-niveaux. Le premier article traitant de cette méthode est proposé par [Donath et Hoffman, 1973]. Il y a de nombreux articles qui présentent l'application de méthode spectrale au partitionnement de graphes ([Pothen *et al.*, 1990][Simon, 1991][Barnard et Simon, 1994][Hendrickson et Leland, 1995]). La méthode spectrale doit son nom au théorème spectral de l'algèbre linéaire. Ce théorème permet d'affirmer la diagonalisation des matrices réelles. Il justifie également la décomposition des matrices symétriques réelles en valeurs propres dans une base orthonormale de vecteurs propres. Le problème de partitionnement de graphe peut être ramené à la résolution d'un système numérique $Mx = \lambda x$. Résoudre ce système numérique consiste à trouver une base orthogonale de vecteurs propres de la matrice M. L'un des inconvénients des cette méthode est qu'elle est gourmande en temps de calcul, et surtout en espace mémoire.

La méthode multi-niveaux est actuellement la méthode la plus utilisée pour résoudre le problème de partitionnement de graphes. Cette méthode est utilisée pour la première fois par [Barnard et Simon, 1994]. Cependant, elle est vite reconnue comme une méthode très efficace et qui apporte une vision plus globale du graphe ([Karypis et Kumar, 1998]) que les méthodes d'expansion de région et la méthodes spectrale. L'idée initiale de cette

méthode est de regrouper les sommets ensemble pour s'occuper de groupes de sommets plutôt que de chaque sommet un par un. Cette méthode se décompose en trois phases bien distinctes et qui agissent chacune sur le graphe G.

- Contraction : la phase de contraction est de nature itérative. À chaque itération, les sommets du graphe résultant de l'itération précédente sont regroupés pour former un graphe similaire, mais le nombre de sommets est plus petit que le précédent. Ainsi, une famille de graphes $\{G_1, ..., G_n\}$, avec $G_1 = G$ est créée, telle que pour chaque graphe G_{i+1}, chacun de ses sommets représente un groupe de sommets du graphe G_i précédent. Le processus s'arrête lorsque le graphe est suffisamment petit.
- Partitionnement : le but de cette étape est de créer une partition P_k^n du graphe G_n. Pour cela, le graphe G_n résultant de l'étape de contraction peut être partitionné en utilisant une heuristique de partitionnement comme par exemple une méthode d'expansion de région.
- Affinage : l'étape d'affinage consiste à projeter le découpage de la partition P_k^n sur le graphe initial G. Cependant, projeter directement P_k^n sur G ne produit pas souvent un découpage de bonne qualité. Pour résoudre ce problème, une famille de partitions $\{P_k^1, ..., P_k^n\}$ est créée et pour laquelle chaque élément P_k^i correspond au partitionnement de P_k^{i+1} projeté sur G_i. Pour améliorer localement la partition finale, chaque partition intermédiaire P_k^i est affinée en utilisant un algorithme local d'affinage de type Kernighan-Lin. La partition finale possède alors les propriétés d'être globalement et localement de bonne qualité.

L'algorithme de [Kernighan et Lin, 1970] permet d'affiner la bissection d'un graphe précédemment obtenu. Il s'agit donc d'un algorithme d'optimisation locale. L'idée principale de cet algorithme est de trouver deux sous-ensembles de sommets de même taille, chacun dans une partie de bissection, tels que leur échange diminue le coût de coupe de bissection. Afin de trouver une partition localement optimale, l'algorithme procède comme suit : partant d'une bissection existante, l'algorithme échange successivement deux sous-ensembles de la bissection jusqu'à plus aucun sous-ensemble diminuant le coût de coupe ne peux être trouvé. La dernière bissection obtenue est donc la bissection de coût de coupe minimal trouvé par l'algorithme. L'algorithme de Kernighan-Lin est d'autant plus rapide et performant si sa partition initiale est de bonne qualité. En effet, si la partition initiale est de bonne qualité, le nombre d'itérations de l'algorithme sera petit car il sera rapidement impossible de trouver des sous-ensembles à échanger. De plus, l'optimisation trouvée est locale. Ainsi, pour avoir un coût de coupe minimal, il est préférable de partir d'une bissection de coût de coupe faible. On peut utiliser la méthode d'expansion de région pour créer la bissection initiale. Pour avoir plus de connaissance sur des méthodes heuristiques pour le problème de partitionnement de graphe, le lecteur est renvoyé à [Bichot et Siarry, 2011].

Concernant les méthodes exactes pour résoudre le problème de partitionnement de graphes, [Brunetta *et al.*, 1997] proposent une méthode par séparation et coupe (branch-and-cut) basé sur une relaxation de programmation linéaire et des coupes basées sur des techniques de séparation. Une approche de génération de colonnes est développée par [Johnson *et al.*, 1993], tandis que [Mitchell, 2001] développe une approche polyédrale. [Sensen, 2001] développe une méthode par séparation et évaluation (branch-and-bound) basée sur une borne inférieure obtenue par la résolution d'un problème de multiflot.

Le problème de partitionnement de graphes et le problème de sectorisation, sont tous pour partitionner une grande zone dans plusieurs plus petites zones. La différence principale entre ces deux problèmes est que dans le problème de sectorisation, les poids des arêtes entre les partitions différentes n'ont pas besoin d'être pris en compte. Et dans le problème de sectorisation, les poids des sommets de chaque partition sont évalués dans une fonction objective.

Chapitre 3

Méthodes heuristiques

Dans ce chapitre, nous présentons les différents problèmes de sectorisation que nous abordons dans cette thèse. Comme ces problèmes sont NP-difficiles, il y a deux catégories générales de méthodes pour les résoudre : des méthodes exactes qui calculent une solution optimale de sectorisation, mais avec une complexité exponentielle, et des méthodes heuristiques qui essaient de calculer rapidement une solution approchée. Avec la contrainte de rapidité imposée souvent par le coté industriel, dans un premier temps, nous nous intéressons à des méthodes heuristiques qui permet de trouver une solution réalisable avec un temps de calcul raisonnable. Plus précisément, une des contraintes imposée par la société Articque est de pouvoir résoudre des problèmes de très grand taille (plusieurs millions d'éléments à sectoriser). Cela implique nécessairement un choix dans les méthodes à utiliser. Dans la section 1.3, nous avons présenté les différents problèmes de sectorisation en multicritère. Dans ce chapitre, nous abordons les problèmes de sectorisation bicritère que nous avons plus particulièrement abordés et nous présentons quelques heuristiques que nous développons.

3.1 Le problème de sectorisation bicritère à partir de pôles

3.1.1 Des heuristiques pour calculer un optimum de Pareto

Ce problème de sectorisation bicritère peut être formellement défini comme suit. Soit $G = (V, E)$ un graphe non-orienté où chaque sommet $v_i \in V$ représente un élément de la carte géographique. Une arête $(v_i, v_k) \in E$ relie deux sommets v_i et v_k du graphe G si et seulement si les éléments associés à v_i et v_k sur la carte sont voisins géographiquement. À chaque sommet $v_i \in V$ on associe un vecteur d'évaluation à deux dimensions $[w_i^1, w_i^2]$ pour représenter ses deux informations statistiques. On connaît le nombre de secteurs p à construire. L'objectif est d'avoir une p-partition de ces éléments en minimisant deux fonctions objectif avec les formes suivantes :

$$(i) \quad Z^1 = \max_j |\sum_{v_i \in S_j} w_i^1 - e_j^1|,$$

$$(ii) \quad Z^2 = \max_{j \neq j'} |\sum_{v_i \in S_j} w_i^2 - \sum_{v_{i'} \in S_{j'}} w_{i'}^2|,$$

avec $\{S_1, ..., S_j, ..., S_p\}$ une partition de V et chaque secteur S_j est connexe dans G. On suppose que e_j^1 est une valeur de référence pour le secteur S_j pour le premier critère. Il est désirable que la valeur du secteur S_j sur le premier critère, notée par $f_j^1 = \sum_{v_i \in S_j} w_i^1$, soit la plus proche possible de e_j^1. On a aussi une contrainte additionnelle qui impose que la valeur de chaque secteur pour le second critère ne peut pas dépasser une valeur prédéfinie par le Décideur, notée δ. Nous notons par δ-contrainte cette contrainte . Dans le problème de sectorisation bicritère à partir de pôles, on suppose par ailleurs qu'à chaque secteur S_j on attache un pôle (centre) $c_j \in V$ qui est défini comme étant un élément géographique donné par avance (par exemple, une ville de départ/arrivée pour un commercial).

Comme nous avons deux critères dans ce problème, dans notre étude, nous utilisons l'approche ϵ-contrainte pour trouver les optima de Pareto. Avec cette approche, nous prenons le critère Z^1 à minimiser, et nous mettons une borne ϵ sur la valeur du critère Z^2. La valeur de ϵ est définie par le Décideur. Toute solution optimale pour ce problème dit ϵ-contraint est un optima de Pareto ([T'kindt et Billaut, 2006]). Donc le problème de sectorisation bicritère revient à trouver une solution qui minimise Z^1 en respectant l'ϵ-contraintes sur Z^2 et les autre contraints décrites ci-dessus, comme la δ-contrainte et la contrainte de connexité. Formellement, l'ϵ-contrainte et la δ-contrainte sont définies comme suit :
ϵ-contrainte

$$|\sum_{v_i \in S_j} w_i^2 - \sum_{v_{i'} \in S_{j'}} w_{i'}^2| < \epsilon, \text{ pour } j, j' \in \{1, ..., p\} \text{ et } j \neq j'$$

δ-contrainte

$$\sum_{v_i \in S_j} w_i^2 < \delta, \text{ pour tout } j \in \{1, ..., p\}$$

Nous avons développé trois heuristiques pour trouver une solution faisable de sectorisation. Ces heuristiques ressemblent aux méthodes d'expansion de région introduites dans la section 2.3 pour résoudre le problème du partitionnement de graphe. Ces heuristiques consistent à créer itérativement des ensembles (secteurs) S_j, $j \in \{1, ..., p\}$, chacun rassemblant une partie des sommets du graphe. Chaque secteur S_j est initialisé avec son pôle c_j. Pendant l'exécution des heuristiques, à chaque itération, nous avons trois types d'ensembles de sommets du graphe : l'ensemble D_j, qui est l'ensemble des sommets qui déjà affectés au secteur S_j ; l'ensemble L_j, qui est l'ensemble des sommets qui peuvent être affectés au secteur S_j à la prochaine itération ; et l'ensemble des sommets restants, noté R. Chaque élément d'un ensemble L_j associé au secteur S_j satisfait ces trois conditions : (1) il n'est pas encore affecté ; (2) il est adjacent à au moins un élément dans l'ensemble D_j ; (3) avec son intégration dans S_j, l'ϵ-contrainte et la δ-contrainte ne serons pas violées. À chaque itération, un sommet dans les ensembles L_j est choisi et il sera ajouté à l'ensemble correspondant D_j. Puis les ensembles L_j seront mis à jour. Le processus s'arrête lorsque tous les ensembles L_j sont vides. À la fin du processus, si les sommets sont tous affectés, alors cette solution est faisable pour le problème de sectorisation bicritère.

Nous nommons ces trois heuristiques par H_1, H_2 et H_R. Ces heuristiques sont toutes glouton et se comportent différemment avec des réponses différentes aux questions suivantes : à chaque itération, (1) quel est le secteur S_j on choisit pour être étendu ; (2)

Heuristique H_1

Étape 1. $\forall j \in \{1, ..., p\}$, $S_j = \{c_j\}$. Construire L_j.

Étape 2. **Tant Que** $\cup_{j\in\{1,...,p\}} L_j \neq \emptyset$, **Répéter**

2.1. Trouver (v_i, S_j) :=arg $\min_{v_l \in L_j}\{\max_{k\in\{1,...,p\}} |e_k^1 - (f_k^1 + w_l^1)|\}$.

2.2. Affecter v_i à S_j, $f_j^1 := f_j^1 + w_i^1$, $f_j^2 := f_j^2 + w_i^2$.

2.3. $\forall k \in \{1, ..., p\}$, retirer v_i de L_k s'il s'y trouvait.

2.4. Ajouter à L_j les éléments connexes à v_i.

Fin Tant Que

Étape 3. $S = \{S_1, ..., S_p\}$, $Z^1 = \max_{1\leq j\leq p} |f_j^1 - e_j^1|$,
$Z^2 = \max_{1\leq j,j'\leq p} |f_j^2 - f_{j'}^2|$. Retourner ($S$, Z^1, Z^2).

FIGURE 3.1 – Heuristique H_1

Heuristique H_2

Étape 1. $\forall j \in \{1, ..., p\}$, $S_j = \{c_j\}$. Construire L_j.

Étape 2. **Tant Que** $\cup_{j\in\{1,...,p\}} L_j \neq \emptyset$, **Répéter**

2.1. Trouver S_j :=arg $\max_{k\in\{1,...,p\}/L_k\neq\emptyset}(e_k^1 - f_k^1)$.

2.2. Trouver v_i :=arg $\max_{v_l\in L_j}(w_l^1)$.

2.3. Affecter v_i à S_j, $f_j^1 := f_j^1 + w_i^1$, $f_j^2 := f_j^2 + w_i^2$.

2.4. $\forall k \in \{1, ..., p\}$, retirer v_i de L_k s'il s'y trouvait.

2.5. Ajouter à L_j les éléments connexes à v_i.

Fin Tant Que

Étape 3. $S = \{S_1, ..., S_p\}$, $Z^1 = \max_{1\leq j\leq p} |f_j^1 - e_j^1|$,
$Z^2 = \max_{1\leq j,j'\leq p} |f_j^2 - f_{j'}^2|$. Retourner ($S$, Z^1, Z^2).

FIGURE 3.2 – Heuristique H_2

quel est l'élément dans L_j choisi pour être ajouté dans le secteur S_j ? Ces heuristiques retournent idéalement une solution faisable S et sa valeur pour les critères Z^1 et Z^2. Ces heuristiques sont décrites dans les figures 3.1, 3.2 et 3.3.

La seule différence entre ces trois heuristiques est dans l'Étape 2. À chaque itération, l'heuristique H_1 choisit un élément v_i dans tous les ensembles L_j, $j \in \{1, ..., p\}$ tel que avec son intégration dans S_j, la valeur de la fonction objectif Z^1 reste la plus petite ; l'heuristique H_2 choisit un secteur S_j dont la liste correspondante L_j n'est pas vide et qui porte l'écart le plus important entre sa valeur actuelle f_j^1 et la valeur de référence e_j^1. Puis elle choisit un élément v_i dans L_j qui a la valeur w_i^1 la plus importante ; l'heuristique H_R choisit un secteur S_j aléatoirement dont la liste correspondant L_j n'est pas vide, puis elle choisit un élément v_i aléatoirement dans L_j.

Pour évaluer la qualité d'une heuristique, une façon est de comparer la qualité de la solution trouvée par cette heuristique par rapport aux solutions optimales. Pour trouver la solution optimale du problème de sectorisation bicritère à partir de pôles, nous développons un modèle mathématique de la programmation linéaire en nombres entiers (PLNE) pour ce problème. Pour résoudre ce modèle de la PLNE, nous utilisons l'outil Ilog Cplex qui est solveur mathématique pour résoudre des problèmes d'optimisation linéaire.

Heuristique H_R

Étape 1. $\forall j \in \{1, ..., p\}$, $S_j = \{c_j\}$. Construire L_j.

Étape 2. **Tant Que** $\cup_{j\in\{1,...,p\}} L_j \neq \emptyset$, **Répéter**

2.1. Sélectionner S_j, $j \in \{1, ..., p\}$, aléatoirement tel que $L_j \neq \emptyset$.

2.2. Sélectionner $v_i \in L_j$ aléatoirement.

2.3. Affecter v_i à S_j, $f_j^1 := f_j^1 + w_i^1$, $f_j^2 := f_j^2 + w_i^2$.

2.4. $\forall k \in \{1, ..., p\}$, retirer v_i de L_k s'il s'y trouvait.

2.5. Ajouter à L_j les éléments connexes à v_i.

Fin Tant Que

Étape 3. $S = \{S_1, ..., S_p\}$, $Z^1 = \max_{1\leq j\leq p} |f_j^1 - e_j^1|$,
$Z^2 = \max_{1\leq j,j'\leq p} |f_j^2 - f_{j'}^2|$. Retourner ($S$, Z^1, Z^2).

FIGURE 3.3 – Heuristique H_R

Nous présentons d'abord un modèle mathématique du problème de sectorisation bicritère à partir de pôles. Pour respecter la contrainte de connexité, il doit y avoir un arbre couvrant tous les sommets d'un secteur dans le graphe G. Soient x_{ij} et y_{ijk} des variables binaires : x_{ij} est égal à 1 si et seulement si v_i est affecté au secteur S_j et y_{ijk} est égal à 1 si et seulement si l'arête (v_i, v_k) est choisie pour construire l'arbre couvrant associé au secteur S_j. Pour exclure des cycles dans un arbre couvrant, des variables supplémentaires u_{ij} sont utilisés. Nous désignons les ensembles d'indices de sommets et de secteurs par $I = \{1, ..., n\}$ et $J = \{1, ..., p\}$ respectivement. Nous avons aussi un ensemble de pôles $C = \{c_1, ..., c_p\}$. Le problème est formulé comme un problème d'affectation avec la contrainte de connexité et le modèle mathématique est donné ci-dessous.

Données

Le nombre de sommets n,
Le nombre de secteurs souhaités p,
La valeur de référence du premier critère associée au secteur S_j : e_j^1,
Les valeurs statistiques associées à chaque sommet v_i : $[w_i^1, w_i^2]$,
L'indice d'adjacence entre deux sommets a_{ik} :

$$a_{ik} = \begin{cases} 1 & \text{s'il existe une arête entre les sommets } v_i \text{ et } v_k \text{ dans le graphe } G, \\ 0 & \text{sinon.} \end{cases}$$

Les valeurs des ϵ et δ.

Variables

x_{ij}, exprime la relation entre le sommet v_i et le secteur S_j :

$$x_{ij} = \begin{cases} 1 & \text{si le sommet } v_i \text{ est affecté au secteur } S_j, \\ 0 & \text{sinon.} \end{cases}$$

$$y_{ijk} = \begin{cases} 1 & \text{si l'arête } (v_i, v_k) \text{ est sélectionnée pour construire} \\ & \text{l'arbre couvrant associé au secteur } S_j \\ 0 & \text{sinon} \end{cases}$$

u_{ij}, la profondeur du sommet v_i dans l'arbre couvrant associé au secteur S_j.

Contraintes

$$\sum_{j=1}^{p} x_{ij} = 1, \quad \forall i = 1, ..., n \tag{3.1}$$

$$x_{c_j,j} = 1, \quad \forall j = 1, ..., p \tag{3.2}$$

$$y_{ijk} \leq x_{ij}, y_{ijk} \leq x_{kj}, y_{ijk} \leq a_{ik}, \quad \forall i, k = 1, ..., n, \quad \forall j = 1, ..., p \tag{3.3}$$

$$\sum_{k=1}^{n} y_{kji} = x_{ij}, \quad \forall i = 1, ..., n, v_i \notin C, \quad \forall j = 1, ..., p \tag{3.4}$$

$$n \cdot y_{ijk} + u_{ij} \leq u_{kj} + (n-1), \quad \forall i, k = 1, ..., n, \quad \forall j = 1, ..., p \tag{3.5}$$

$$u_{ij} \leq n x_{ij}, u_{ij} \geq x_{ij} \quad \forall i = 1, ..., n, \quad \forall j = 1, ..., p \tag{3.6}$$

$$\sum_{i=1}^{n} w_i^2 x_{ij} \leq \delta, \quad \forall j = 1, ..., p \tag{3.7}$$

$$x_{ij}, y_{ijk} \in \{0, 1\}, u_{ij} \in \{1, ..., n\} \quad \forall i = 1, ..., n, \forall j = 1, ..., p \tag{3.8}$$

Fonction Objectif

$$\text{Minimiser} \max_{1 \leq j \leq p} |\sum_{i=1}^{n} w_i^1 x_{ij} - e_j^1| \tag{3.9}$$

$$\text{Minimiser} \max_{1 \leq j, j' \leq p, j \neq j'} |\sum_{i=1}^{n} w_i^2 x_{ij} - \sum_{i=1}^{n} w_i^2 x_{ij'}| \tag{3.10}$$

Les contraintes (3.1) sont des contraintes d'affectation et elles assurent que chaque élément est affecté à exactement un secteur. Les contraintes (3.2) indiquent qu'il existe un pôle c_j dans chaque secteur S_j. Les contraintes (3.3) déclarent qu'une arête (v_i, v_k) ne peut pas être dans l'arbre couvrant du secteur S_j si v_i et v_k ne sont pas affectés au secteur S_j où si v_i et v_k ne sont pas voisins. Les contraintes (3.4) indiquent qu'il ne peut pas exister plus qu'une arête entrant pour chaque sommet dans un arbre couvrant. Les contraintes (3.5) et (3.6) sont les conditions d'élimination de cycles dans un arbre : elles sont dérivées du problème du voyageur de commerce telles qu'énoncé par [Miller *et al.*, 1960]. Les contraintes (3.7) sont les δ-contraintes qui imposent que la valeur de chaque secteur pour le second critère ne peut pas dépasser une valeur δ. Les contraintes (3.8) sont des contraintes d'intégralité des variables de décision.

Nous avons deux critères (3.9) et (3.10) à minimiser. Nous considérons le critère (3.9) comme le critère principal, et nous utilisons l'ϵ-contrainte pour borner le deuxième critère. Le critère (3.10) est donc remplacé par la contrainte :

$$\max_{1\leq j,j'\leq p, j\neq j'} |\sum_{i=1}^{n} w_i^2 x_{ij} - \sum_{i=1}^{n} w_i^2 x_{ij'}| \leq \epsilon \tag{3.11}$$

Les trois heuristiques H_1, H_2, H_R, et le modèle mathématique intégré dans Cplex, ont été testé sur des instances aléatoires que nous avons générées. Nous présentons ici la façon dont nous générons ces instances.

Une carte géographique, comme la carte de la France en département, peut être considérée comme un graphe planaire ([Berge, 1973]). Pour créer un graphe planaire, nous générons d'abord un ensemble de n points dans un plan, avec chaque point $p_i = (x_i, y_i)$, $i \in \{1, ..., n\}$, (x_i, y_i) est les coordonnées dans le plan du point p_i. Ces points peuvent être considérés comme les barycentres des éléments géographiques. À partir de ces points, nous utilisons la triangulation de Delaunay ([Berg *et al.*, 2008]) pour nous fournir un ensemble de triangles qui pourront ensuite être utilisés pour construire une matrice de relations d'adjacence.

La triangulation de Delaunay est une triangulation particulière. Une triangulation d'un ensemble de points P dans le plan est une triangulation de l'enveloppe convexe de P, tous les points de P formant des sommets de cette triangulation. La triangulation de Delaunay d'un ensemble de points P du plan est une triangulation $DT(P)$ telle qu'aucun point de P n'est à l'intérieur du cercle circonscrit d'un des triangles de $DT(P)$. Un exemple de la triangulation de Delaunay avec des cercles circonscrits des triangles est présenté dans la Figure 3.4. Une chose importante est qu'il n'existe pas de triangulation de Delaunay pour un ensemble de points alignés. Le graphe de triangulation de Delaunay d'un ensemble de points possède un certain nombre de propriétés. Une propriété est qu'il est toujours un graphe planaire : un graphe qui a la particularité de pouvoir se représenter sur un plan sans qu'aucune arête n'en croise une autre. La triangulation de Delaunay a aussi une relation avec le diagramme de Voronoï.

Le diagramme de Voronoï d'un ensemble de n points P est la subdivision du plan en n régions, une pour chaque point de P, de sorte que la région d'un point $p \in P$ contient tous les points dans le plan pour lequel p est le point le plus proche. La région d'un point p est appelé la cellule de Voronoi de p. Un exemple du diagramme de Voronoï avec des cellules de Voronoi est dans la Figure 3.5.

La triangulation de Delaunay d'un ensemble discret P de points est le graphe dual du diagramme de Voronoï associé à P. À chaque cellule du diagramme de Voronoï est associée un sommet dans la triangulation de Delaunay. Ces sommets sont reliés entre eux par une arête si les cellules associées sont voisines. Les arêtes du diagramme de Voronoï sont sur les médiatrices des arêtes de la triangulation de Delaunay. Voir un exemple présenté dans la Figure 3.6.

Pour générer un graphe représentant un terrain géographique, nous allons procéder en trois étapes : (1) Nous générons n points aléatoirement sur le plan. Ces points sont équivalents à des sommets dans le graphe. Un point peut être considéré comme le centre d'un élément géographique à sectoriser et le nombre de point n est aussi le nombre d'éléments ; (2) À partir des points générés, nous pouvons générer la triangulation de Delau-

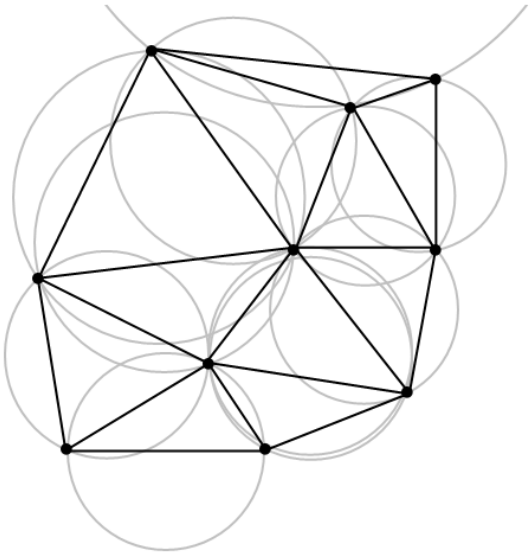

FIGURE 3.4 – Une triangulation de Delaunay avec les cercles circonscrits

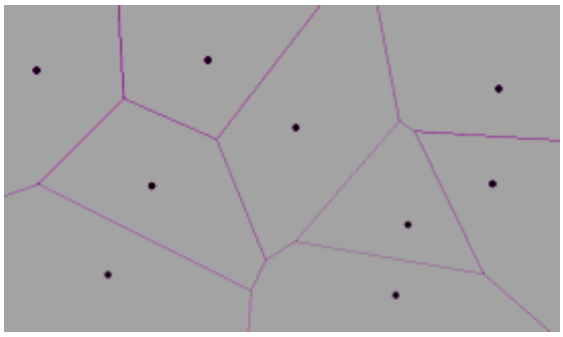

FIGURE 3.5 – Exemple de diagramme de Voronoï

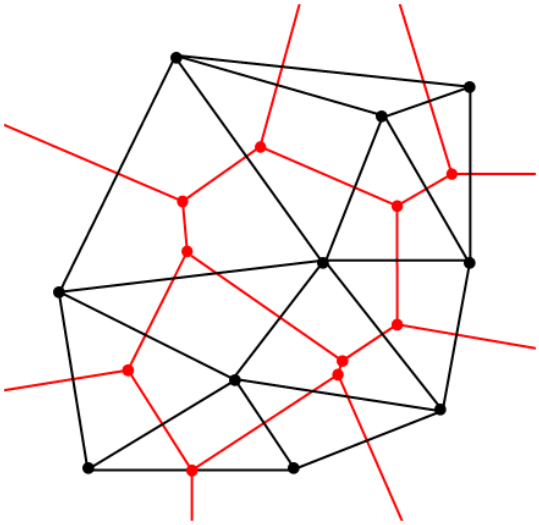

FIGURE 3.6 – Superposition d'un diagramme de Voronoï (en traits rouges) et de sa triangulation de Delaunay (en traits noirs)

nay. Il existe beaucoup d'algorithmes dans la littérature pour calculer la triangulation de Delaunay, comme *les algorithmes de basculement*, *d'incrémentation* et *diviser pour régner* ([Berg *et al.*, 2008]). L'algorithme *diviser pour régner* est le plus efficace avec une complexité en $O(nlog(n))$. Nous avons choisi un algorithme simple qui est présenté dans [Guibas *et al.*, 1992]. Cet algorithme est incrémental aléatoirement. Il ajoute les points dans un ordre aléatoire et il maintient une triangulation de Delaunay de l'ensemble des points déjà considérés. La complexité de cet algorithme est en $O(n^2)$ mais il est plus facile à comprendre et à implémenter ; (3) Basé sur la triangulation calculée, nous pouvons obtenir une matrice d'adjacence qui est représentée par les valeurs de a_{ik}. Nous mettons $a_{ik} = 1$ s'il existe une arête entre deux points (sommets) p_i et p_k.

Dans une instance du problème de sectorisation, nous avons d'autres données à générer. Les p pôles sont générés aléatoirement parmi les n éléments. Les valeurs statistiques $[w_i^1, w_i^2]$ associées à chaque sommet v_i sont deux entiers aléatoires entre 1 et 100. La valeur de référence pour le premier critère associées à chaque secteur S_j est obtenue par : $e_j^1 = \sum_{v_i \in G} w_i^1/p$. Nous plaçons donc dans le cas de secteurs équilibrés Quand un nombre d'éléments n est fixé, nous avons trois valeurs pour le nombre de secteurs (pôles) p : $\lceil 25\% \times n \rceil$, $\lceil 50\% \times n \rceil$ et $\lceil 75\% \times n \rceil$.

Nous testons d'abord les heuristiques H_1, H_2, H_R et Cplex Tronqué. Pour Cplex Tronqué, nous donnons au solveur Cplex 15 minutes pour résoudre le modèle mathématique. Si dans ce temps limité, Cplex arrive à trouver une solution faisable, nous enregistrons cette solution S et sa valeur du critère Z^1. Ces heuristique sont lancées sur des instances dont le nombre d'éléments n égale à 100, 300, 500, 1000 et 1500. Pour chaque couple (n, p), nous générons 30 instances. Nous fixons les valeurs de ϵ et δ de la façon suivante : $\epsilon = 300\% \times \sum_{v_i \in G} w_i^1/p$, $\delta = 350\% \times \sum_{v_i \in G} w_i^1/p$. Quand les valeurs de ϵ et δ sont plus petites, ces heuristiques ont moins de chance pour trouver une solution faisable.

n	p	$\%Infai$	$Temps_{max}$	$Temps_{moy}$	Dev_{min}	Dev_{max}	Dev_{moy}
100	25	10.0	0.02	0.00	20.08	182.89	95.25
	50	10.0	0.02	0.00	0.00	175.55	42.25
	75	0.0	0.02	0.00	0.00	154.66	12.90
300	75	30.0	0.05	0.03	26.08	154.47	95.23
	150	30.0	0.05	0.04	0.00	96.19	41.59
	225	10.0	0.05	0.02	0.00	140.00	14.15
500	125	46.7	0.13	0.11	2.53	169.04	97.22
	250	23.3	0.17	0.15	6.42	131.30	62.40
	375	20.0	0.09	0.08	0.00	53.22	9.87
1000	250	56.7	0.82	0.75	50.76	174.12	101.09
	500	53.3	1.15	1.11	0.00	145.54	58.74
	750	13.3	0.63	0.58	0.00	39.21	2.77
1500	375	53.3	2.52	2.40	68.80	146.01	102.27
	750	93.3	3.76	3.60	0.00	71.84	38.48
	1125	40.0	2.03	1.84	0.00	64.48	4.48

TABLE 3.1 – *Comparaisons de H_1 avec les autres heuristiques*

Les critères que nous utilisons pour évaluer ces heuristiques sont : (1) le pourcentage d'infaisabilité $Infai$ évalue le pourcentage d'instances pour lesquelles une heuristique n'a pas réussi à trouver une solution faisable, la raison peut être que les instances sont infaisable à cause de l'ϵ-contrainte ou de la δ-contrainte ; (2) le temps maximal T_{max} et le temps moyen T_{moy} de calcul ; (3) la déviation minimale Dev_{min}, la déviation maximale Dev_{max} et la déviation moyenne Dev_{moy}. La déviation (en pourcentage) est définie par $((Z^1_H - Z^1_{Best})/Z^1_{Best})\times 100$, avec Z^1_{Best} la valeur du critère Z^1 de la meilleure solution trouvée parmi les heuristiques, et Z^1_H la valeur du critère Z^1 de la solution trouvée par une heuristique H. Les déviations pour l'heuristique H sont calculées uniquement sur les instances qu'elle résout. Les tests expérimentaux ont été réalisés sur une machine équipée d'un processeur Pentium Core 2 Duo 3.0Ghz et de 2.48G de RAM. Nous avons utilisé le langage C++ et la librairie standard (STL) pour développer les heuristiques.

Comme Cplex Tronqué n'arrive pas à trouver une solution faisable en 15 minutes pour les instances dont la taille est supérieure à 100, nous ne montrons les résultats que pour les heuristiques H_1, H_2 et H_R. Cela illustre la difficulté des instances générées, au moins pour un solveur mathématique. Dans les tableaux 3.2, 3.3 et 3.4, nous pouvons remarquer que H_2 trouve plus souvent une solution faisable que H_1 et H_R. Au niveau du temps d'exécution, ces trois heuristiques sont presque pareilles. Nous remarquons aussi que, à partir des colonnes de déviations, la qualité des solutions trouvées par H_2 est bien meilleure que celles trouvées par H_1 et H_R.

Nous avons également cherché à évaluer les résolutions exactes du modèle mathématique par Cplex 10.2 (sans limite de temps de calcul) en de petites instances dont la taille (n) varie entre 6 et 18. Pour chaque couple (n, p), nous générons toujours 30 instances. Nous fixons les valeurs de ϵ et δ comme : $\epsilon = 100\% \times \sum_{v_i \in G} w^1_i/p$ et $\delta = 150\% \times \sum_{v_i \in G} w^1_i/p$. Nous remarquons dans le tableau 3.4 que Cplex commence à avoir du mal pour résoudre les instances de taille 18 avec un nombre de secteurs égale à 5. Cplex utilise plus de 20 minutes

n	p	$\%Infai$	$Temps_{max}$	$Temps_{moy}$	Dev_{min}	Dev_{max}	Dev_{moy}
100	25	0.0	0.02	0.00	0.00	0.00	0.00
	50	0.0	0.02	0.00	0.00	34.94	1.77
	75	0.0	0.02	0.00	0.00	49.05	2.66
300	75	0.0	0.05	0.03	0.00	0.00	0.00
	150	3.3	0.05	0.04	0.00	0.00	0.00
	225	6.7	0.05	0.02	0.00	48.89	2.58
500	125	0.0	0.14	0.12	0.00	0.00	0.00
	250	0.0	0.17	0.16	0.00	0.00	0.00
	375	3.3	0.09	0.08	0.00	0.00	0.00
1000	250	3.3	0.89	0.83	0.00	0.00	0.00
	500	0.0	1.19	1.13	0.00	0.00	0.00
	750	3.3	0.63	0.58	0.00	16.35	0.56
1500	375	3.3	2.78	2.65	0.00	0.00	0.00
	750	16.7	3.89	3.66	0.00	0.00	0.00
	1125	36.7	1.98	1.89	0.00	0.00	0.00

TABLE 3.2 – *Comparaisons de H_2 avec les autres heuristiques*

n	p	$\%Infai$	$Temps_{max}$	$Temps_{moy}$	Dev_{max}	Dev_{moy}	Dev_{min}
100	25	36.7	0.02	0.00	89.30	685.26	315.27
	50	30.0	0.02	0.00	51.64	429.16	147.67
	75	0.0	0.02	0.00	23.23	220.00	110.21
300	75	73.3	0.05	0.03	153.04	358.23	269.90
	150	40.0	0.05	0.04	62.90	403.52	159.00
	225	6.7	0.05	0.02	0.00	244.94	67.07
500	125	73.3	0.14	0.13	243.55	476.37	312.49
	250	33.3	0.18	0.16	37.62	306.45	192.59
	375	23.3	0.09	0.08	0.00	185.48	71.67
1000	250	100.0					
	500	76.7	1.23	1.16	56.12	342.45	216.63
	750	20.0	0.64	0.58	0.00	189.21	45.54
1500	375	100.0					
	750	80.0	3.93	3.76	0.00	237.86	127.37
	1125	43.4	2.04	1.86	0.00	185.04	35.11

TABLE 3.3 – *Comparaisons de H_R avec les autres heuristiques*

n	p	$\%Infai$	$Temps_{max}$	$Temps_{moy}$
6	2	0.0	0.031	0.017
	3	6.7	0.031	0.014
	5	26.7	0.016	0.011
10	3	0.0	0.172	0.091
	5	3.3	0.109	0.050
	8	30.0	0.047	0.032
14	4	0.0	0.750	0.229
	7	0.0	0.328	0.172
	11	30.0	0.078	0.047
16	4	0.0	705.906	86.027
	8	0.0	0.703	0.335
	12	16.7	0.109	0.100
18	5	0.0	1245.875	191.326
	9	0.0	1.094	0.623
	14	60.0	0.156	0.138

TABLE 3.4 – *Résultats avec Cplex*

sur certaines instances et plus de 3 minutes en moyenne sur ces instances. Généralement, pour des instances dont la taille est fixée, quand p augmente, Cplex utilise moins de temps. Néanmoins, le taux de l'infaisabilité est plus élevé à cause de l'ϵ-contrainte et de la δ-contrainte.

Comme H_2 est meilleure que les autres heuristiques présentées, nous la testons aussi sur ces petites instances pour l'évaluer par rapport à la solution optimale. Dans le tableau 3.5 nous présentons les résultats obtenus en terme de pourcentage de solution optimale trouvée par H_2. Nous remarquons que quand la taille de l'instance augmente, H_2 trouve moins souvent la solution optimale. Et généralement, pour des instances dont la taille est fixée, quand p augmente, H_2 trouve plus souvent la solution optimale.

3.1.2 Un algorithme de boîte pour trouver une représentations du front de Pareto

Dans la section 3.1.1, pour résoudre le problème de sectorisation bicritère à partir de pôles, nous utilisons l'approche ϵ-contrainte qui minimise le premier critère Z^1 en bornant supérieurement le deuxième critère Z^2 avec une valeur de ϵ prédéfinie par le Décideur. Avec chaque valeur de ϵ, nous essayons de trouver une solution approchée d'un optimum de Pareto. Dans cette section, nous ne cherchons plus une unique solution de Pareto, mais plusieurs et qui doivent permettre d'échantillonner le front de Pareto.

Nous nous intéressons au calcul d'une représentation du font de Pareto puisque le problème de trouver l'ensemble des solutions Pareto optimales peut être inapplicable en pratique pour des problèmes NP-difficile d'optimisation bicritères ([Ehrgott, 2005]). Nous utilisons un algorithme des boîtes qui est proposé par [Hamacher *et al.*, 2007] pour trouver une représentation du front de Pareto de notre problème de sectorisation bicritère.

n	p	$\%MeilleureSolution$
6	2	53.33
	3	50.00
	5	100.00
10	3	16.67
	5	20.69
	8	100.00
14	4	13.33
	7	10.00
	11	95.24
16	4	3.33
	8	20.00
	12	68.00
18	5	0.00
	9	3.33
	14	100.0

TABLE 3.5 – *Comparaisons entre H_2 et Cplex*

Nous rappelons tout d'abord les notations d'un problème d'optimisation bicritère qui ont été présentées dans la section 1.5.1 et nous ajoutons quelque notations.

Un problème d'optimisation discret bicritère peut être défini comme suit :

$$\min Z(x) = [Z_1(x); Z_2(x)]^T$$

sous

$$x \in S,$$

où x est une solution faisable et S est l'ensemble des solutions faisables. $Z(x)$ dénote un vecteur de critères correspondant à la solution x et $Z(S)$ dénote l'ensemble de vecteurs de critères atteignables. Nous notons S_E l'ensemble de solutions non-dominées et Z_E l'ensemble des vecteurs de critères correspondant à l'ensemble S_E.

Une représentation, *Rep*, est un ensemble de points de $Z(S)$ qui est calculée au lieu de tout l'ensemble non-dominées S_E : on a donc $Rep \subset Z_E$. Comme en général $Rep \neq Z_E$, nous avons besoin d'évaluer la qualité d'une représentation. Il existe plusieurs critères pour mesurer la qualité d'une représentation, dont notamment l'erreur de représentation. L'erreur de représentation est une mesure du plus grand écart entre *Rep* et Z_E. Elle est définie par $ER = \max_{x_1 \in Rep}(\min_{x_2 \in Z_E}||x_1 - x_2||)$ avec $||\cdot||$ une métrique comme par exemple la métrique Euclidienne.

[Hamacher *et al.*, 2007] présentent deux algorithmes des boîtes. Un algorithme des boîtes est une méthode générique pour calculer une représentation du front de Pareto de problèmes discrets d'optimisation bicritères. Une propriété de cette méthode est qu'elle permet de construire des représentations pour lesquelles $ER = 0$, si on sait calculer un optimum de Pareto de façon exacte. Dans la suite, nous allons présenter d'abord la méthode ϵ-contrainte lexicographique qui est une variante de la méthode ϵ-contrainte. Ensuite, une version de l'algorithme des boîtes va être décrite et adaptée pour notre problème de sectorisation.

Le problème ϵ-contrainte lexicographique P_ϵ se présente sous la forme suivante :

$$\text{lex } \min Z(x) = [Z_1(x); Z_2(x)]^T$$

sous

$$Z_2(x) \leq \epsilon \text{ et } x \in S$$

Dans P_ϵ le vecteur $[Z_1(x); Z_2(x)]^T$ est minimisé lexicographiquement, et son ensemble faisable est l'ensemble faisable du problème d'optimisation bicritère avec une contrainte additionnelle bornant la valeur de $Z_2(x)$.

Le problème ϵ-contrainte lexicographique a plusieurs propriétés qui sont très utiles pour construire une représentation de l'ensemble non-dominé. En particulier, il existe une correspondance un à un entre des solutions optimaux de P_ϵ et des solutions non-dominées S_E : une solution optimale de P_ϵ est forcément dans l'ensemble S_E ; inversement, toute solution non-dominée peut être obtenue en optimisant le problème de ϵ-contrainte lexicographique avec un choix adéquat de ϵ. Cette propriété est prouvée par [Hamacher *et al.*, 2007].

Nous décrivons les idées principales de l'algorithme des boîtes avant d'expliquer les détails. Initialement, nous calculons les deux solutions optimales lexicographiques. Ces points déterminent un rectangle (la boîte initiale) contenant l'ensemble complet des vecteurs non-dominés. Dans la suite nous résolvons itérativement P_ϵ avec des valeurs choisies pour ϵ. Avec des nouveaux points non-dominés, un des rectangles va être divisé en plusieurs petits rectangles. Cela donne une collection de rectangles (boîtes) contenant l'ensemble des vecteurs non-dominés à chaque étape de l'algorithme. En plus, pour chaque boîte, on connaît un point non-dominé représentant l'ensemble des points non-dominés à l'intérieur de la boîte. L'algorithme des boîtes s'arrête lorsque une précision, prédéterminée et mesurée par la surface la plus grande des boîtes restantes, est atteinte.

De façon plus détaillée, l'algorithme des boîtes fonctionne comme suit. Initialement, nous déterminons les deux minimas lexicographiques du problème d'optimisation bicritère

$$z_1 = [z_1^1; z_1^2]^T = \text{lex min}_{x \in S}[Z_1(x); Z_2(x)]^T \text{ et}$$

$$z_2 = [z_2^1; z_2^2]^T = \text{lex min}_{x \in S}[Z_2(x); Z_1(x)]^T.$$

En fait, Z_E est un sous-ensemble de $R = R(z_1, z_2)$, qui est le rectangle dans l'espace des critères de sommet supérieur gauche z_2 et sommet inférieur droit z_1 (Figure 3.8). Nous utilisons la notation $R(y_1, y_2)$ pour désigner le rectangle ayant $y_2 = [y_2^1; y_2^2]^T \in \mathbb{R}^2$ comme sommet supérieur gauche et $y_1 = [y_1^1; y_1^2]^T \in \mathbb{R}^2$ comme sommet inférieur droit. De plus, $a(R(y_1, y_2)) = (y_2^1 - y_1^1) \times (y_1^2 - y_2^2)$ dénote le surface du rectangle $R(y_1, y_2)$.

Initialement, la boîte $R = R(z_1, z_2)$ contient l'ensemble complet des vecteurs non-dominés Z_E. Chaque fois qu'un vecteur non-dominé additionnel devient connu pendant l'exécution de l'algorithme, une des boîtes sera divisée en plusieurs plus petits rectangles en conservant tout l'ensemble de Z_E. Nous arrêterons l'algorithme si la surface de la plus grande de ces boîtes est plus petite qu'une certaine précision donnée $\Delta > 0$. Le point inférieur droite de chacun des rectangles est un point représentant et est ajouté à la représentation *Rep* en cours de construction.

Nous présentons ici un algorithme des boîtes qui est proposé par [Hamacher *et al.*, 2007] et appelé 'algorithme a priori'. Dans cet algorithme, Δ est donné comme entrée. Puisque

Un algorithme de boîte
Entrée : $\Delta > 0$.
Sortie : Une représentation $Rep \subseteq S_E$ de précision Δ.

1. $Rep \leftarrow \emptyset$.
2. Calculer les minimas lexicographiques z_1 et z_2, et $a(R(z_1, z_2))$.
3. $Rep \leftarrow \{z_1, z_2\}$.
4. $k \leftarrow \lceil a(R(z_1, z_2))/\Delta \rceil - 1$.
5. **Pour** $j = k...1$ **Faire**
6. $\quad \epsilon_j = \lfloor z_2^2 + j/(k+1)(z_1^2 - z_2^2) \rfloor$.
7. $\quad$ Résoudre P_{ϵ_j}.
8. $\quad Rep \leftarrow Rep \cup \{\hat{z}_j\}$.
9. **Fin pour**

FIGURE 3.7 – Un algorithme de boîte

Δ est le mesure de la taille des boîtes, nous pouvons laisser sa détermination au Décideur. La valeur de Δ peut être, par exemple, défini comme un pourcentage de la surface de la boîte initiale. Nous calculons un certain nombre de valeurs équidistantes pour ϵ sur la base de la valeur de Δ :

$$\epsilon_j = \lfloor z_2^2 + \frac{j}{k+1}(z_1^2 - z_2^2) \rfloor, \quad \forall j = 1, ..., k.$$

$k = \lceil a(R(z_1, z_2))/\Delta \rceil - 1$. Notons $\hat{z}_j$ le vecteur de solution optimale de P_{ϵ_j}. Les points $\hat{z}_j, j = 1, ..., k$, ne sont pas nécessairement distincts. Pour $j = k-1, ..., 1$, nous définissons $p_j = (\epsilon_j + 1, \hat{z}_j^2 - 1)$ si $\hat{z}_{j+1} \neq \hat{z}_j$, sinon $p_j = \hat{z}_j$. Nous mettons $p_k = (\epsilon_k + 1, \hat{z}_k^2 - 1)$. Les points p_j, $j = 1, ..., k$, sont utilisés pour définir les rectangles, mais ils ne sont pas nécessairement faisables dans $Z(S)$. La Figure 3.8 illustre ces définitions. Cet algorithme des boîtes est décrit dans la Figure 3.8.

Nous notons que les problèmes ϵ-contrainte lexicographiques sont résolus de droite à gauche (voir l'étape 5 de l'algorithme). C'est en pratique plus rapide que le sens de gauche à droite, puisque la solution du problème de ϵ-contrainte lexicographique pour $\epsilon = \epsilon_j$ peut être optimal pour $\epsilon_{j-1}, ..., \epsilon_{j-i}$, $i \geq 1$. Par conséquent, la solution de certains problèmes $P_{\epsilon_{j-1}}, ..., P_{\epsilon_{j-i}}$ peut être ignorée.

Nous appliquons cet algorithme des boîtes au problème de sectorisation bicritère avec pôles fixés pour trouver une représentation du front de Pareto. Nous le testons sur les mêmes instances aléatoires générées que pour les heuristiques H_1, H_2, H_R. Nous testons sur 30 instances dont le nombre d'éléments n est égale à 100 et le nombre de secteurs prend les valeurs p : $\lceil 25\% \times n \rceil$, $\lceil 50\% \times n \rceil$ et $\lceil 75\% \times n \rceil$. Comme le problème de sectorisation pour cette taille ne peut pas être résolu par une méthode exacte pour l'instant, nous utilisons l'heuristique H_2 pour résoudre chaque P_{ϵ_j}. Sur chaque instance, nous testons avec quatre tailles différentes de l'intervalle entre ϵ_j et ϵ_{j+1} $(j = k-1, ..., 1)$: $\epsilon_{j+1} - \epsilon_j = \epsilon_1 - z_1^1 = z_2^1 - \epsilon_k \in \{1; 10\% \times (z_2^1 - z_1^1); 25\% \times (z_2^1 - z_1^1); 50\% \times (z_2^1 - z_1^1)\}$. Par ailleurs, et uniquement pour la simplification, Nous fixons δ à une très grande valeur pour ignorer la δ-contrainte. Comme nous utilisons une heuristique pour résoudre chaque P_{ϵ_j}, nous devons à la fin vérifier si chaque solution trouvée est dominée par les autres solutions calculées.

L'objectif de ces tests est d'évaluer le nombre de solutions non-dominées trouvées par

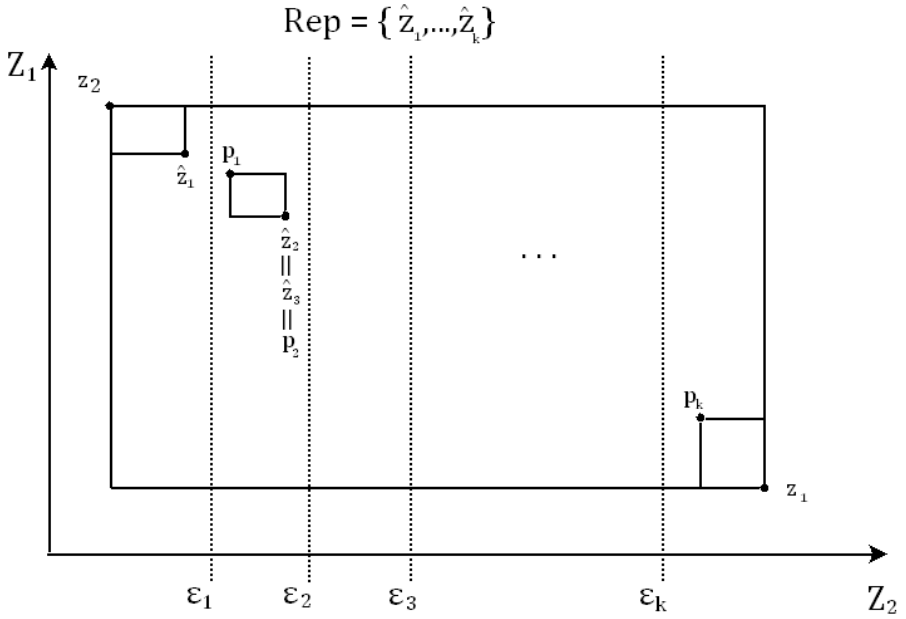

FIGURE 3.8 – La représentation après avoir résolu P_{ϵ_1},P_{ϵ_2},...,P_{ϵ_k}.

intervalle	n	p	#solutions calculées	#solutions non-dominées
1	100	25	160.83	1.37
		50	57.33	1.47
		75	32.97	1.70
10%		25	7.27	1.27
		50	6.23	1.40
		75	8.20	1.67
25%		25	3.77	1.30
		50	3.47	1.30
		75	4.01	1.63
50%		25	2.53	1.23
		50	2.73	1.27
		75	2.70	1.67

TABLE 3.6 – *Un algorithme des boîtes sur le problème de sectorisation*

l'algorithme des boîtes avec l'heuristique H_2 pour notre problème de sectorisation. Les critères que nous utilisons pour évaluer sont : (1) le nombre de solutions que nous calculons, chacune correspondant à un problème P_{ϵ_j} ; (2) le nombre de solutions non-dominées que nous trouvons entre z_1 et z_2. Dans le tableau 3.6, nous pouvons remarquer que cet algorithme des boîtes ne trouve pas beaucoup de solutions non-dominées entre z_1 et z_2 (le nombre en moyenne est entre 1 et 2). Parmi les solutions que nous calculons pour P_{ϵ_j}, la plupart d'entre elles sont dominées par une autre solution non-dominée. La valeur de précision Δ qui est liée avec la taille de l'intervalle entre ϵ_j et ϵ_{j+1} ne joue pas un rôle important sur le nombre de solutions non-dominées trouvées pour le problème de sectorisation.

3.2 Le problème de sectorisation bicritère sans pôle prédéfini

3.2.1 Liens avec le problèmes du localisation-allocation

Dans la section précédente, nous avons présenté le problème de sectorisation bicritère à partir de pôles donnés. Dans ce type de problème de sectorisation, pour chaque secteur S_j, on attache un pôle (centre) $c_j \in V$ qui est défini comme étant un élément géographique donné par avance. Nous avons proposé quelque heuristiques qui traitent ces pôles comme des racines de secteurs (secteurs initiaux) pour résoudre le problème. Dans cette section, nous étudions un autre type de problème de sectorisation bicritère. Dans ce problème, nous n'avons pas de pôle prédéfini, et nous pouvons sectoriser une carte géographique directement en connaissant le nombre de secteurs p et en cherchant éventuellement à localiser des pôles.

Pour définir le problème de sectorisation bicritère sans pôle, nous pouvons reprendre la définition pour le problème de sectorisation bicritère à partir de pôles et nous enlevons juste la définition des pôles. Dans cette section nous nous focalisons sur les liens existants entre les problèmes de localisation-allocation tels qu'introduits dans la section 2.2 et le problème de sectorisation mono-critère avec $Z^1 = \max_j |\sum_{v_i \in S_j} w_i^1 - e_j^1|$. Notre objectif est donc d'étudier si le problème de sectorisation monocritère peut être vu comme étant un sous-problème d'un des problèmes de localisation-affectation.

Les problèmes de localisation-allocation consistent à chercher comment servir au mieux, selon l'objectif, un ensemble de points de demandes (clients) à partir d'un ensemble de sites d'activités. Il existe de nombreux modèles de localisation-allocation qui sont largement étudiés (sous les noms de p-médian, p-centre...). Entre le problème de sectorisation monocritère sans pôle prédéfini et les problèmes de localisation-allocation, nous remarquons que : des éléments basiques dans le problème de sectorisation peuvent être considérés comme des points de demandes dans les problèmes de localisation-allocation ; des pôles à définir dans le problème de sectorisation peuvent être considérés comme des sites d'activité à localiser ; les informations statistiques de chaque élément basique peuvent être considérés comme des poids des points de demandes ; une solution issue de modèles de localisation-allocation selon certaines règles d'allocation peut être une solution connexe pour le problème de sectorisation.

Dans les problèmes de localisation-allocation, chaque point de demandes est souvent affecté à l'activité ouverte la plus proche (règle d'allocation au plus proche) pour des raisons

économiques. Généralement, une solution issue de modèles de localisation-allocation en respectant cette règle d'allocation peut être une solution de forme convexe, connexe, et de "bonne forme" pour le problème de sectorisation.

Dans la suite, nous pouvons prouver que la règle d'allocation au plus proche peut garantir la connexité dans un réseau. Pour justifier cette propriété, nous devons prouver qu'il n'existe pas de point de demandes (ou un bloc de points de demandes) qui est isolé. Un point de demande v_0 qui est affecté à une activité a_0 est dit "isolé" si tous ses voisins sont dans un de ces deux cas : (1) un point de demandes qui est affecté à une activité différente de a_0 ; (2)est une activité différente de a_0. Dans un réseau discret, nous utilisons un graphe pour le représenter. Des points de demandes et des sites d'activités sont généralement sur des sommets de graphe. Pour simplifier la preuve, nous supposons que pour chaque point de demandes, il n'existe qu'une activié "la plus proche" (au sens d'une métrique). Nous notons d_{ij} pour indiquer la distance en terme de plus court chemin entre le sommet v_i et le sommet v_j.

Proposition 1 *Si tous les points de demandes sont affectées à l'activité la plus proche, il n'y aura pas de point de demande "isolé".*

Preuve. Dans un graphe, supposons qu'il existe un point de demandes v_0 qui est isolé, et qui est affecté à l'activité la plus proche a_0. Tous ses sommets adjacents v_1, v_2, v_3,... sont affectés à des activités a_1, a_2, a_3,... respectivement qui sont différentes de a_0. La Figure 3.9 propose un exemple. Comme a_0 est l'activité plus proche pour v_0, nous avons : $d_{v_0a_0} < d_{v_0a_1}$; $d_{v_0a_0} < d_{v_0a_2}$; $d_{v_0a_0} < d_{v_0a_3}$; .... Comme le plus court chemin entre v_0 et a_0 doit passer par un des voisins de v_0, noté v_1, (sinon v_0 est adjacent avec a_0 et ça implique que v_0 n'est pas isolé), nous avons $d_{v_0a_0} = min\{d_{v_0v_1} + d_{v_1a_0}, d_{v_0v_2} + d_{v_2a_0}, d_{v_0v_3} + d_{v_3a_0}, ...\}$. Comme $d_{v_0a_0} < d_{v_0a_1}$, nous avons $d_{v_0v_1} + d_{v_1a_0} < d_{v_0a_1} \leq d_{v_0v_1} + d_{v_1a_1}$, donc $d_{v_1a_0} < d_{v_1a_1}$, ce qui est une contradiction avec a_1 est l'activité la plus proche de v_1. □

Nous prenons un petit exemple dans la Figure 3.10. Dans ce graphe, les sommets a et b sont les sites d'activités et les autre sommets sont les points de demandes. Les sommets c, d, f sont affectés à l'activité a, et le sommet e est affecté à l'activité b. Le sommet e est isolé. Le plus court chemin entre e et b doit passer par un sommet parmi c, d, et f, supposons par le sommet c. Comme $d_{eb} < d_{ea}$, nous avons $d_{eb} = d_{ec} + d_{cb} < d_{ea} \leq d_{ec} + d_{ca}$, donc $d_{cb} < d_{ca}$. Donc, b est plus proche que a de c, et nous avons la contradiction.

Dans le cas d'un espace réel, la propriété de connexité peut être garantie par la partition Voronoï ([Berman *et al.*, 2009]). Même si la règle d'allocation la plus proche peut garantir la connexité dans un réseau et dans un espace réel, elle est plus forte que la contrainte de connexité pour le problème de sectorisation. C'est à dire que cette règle d'allocation ne peut pas garantir l'optimalité pour le problème de sectorisation. Dans le problème de sectorisation, nous nous intéressons aussi à trouver une solution qui est équilibrée.

Nous prenons un petit exemple dans la Figure 3.11 pour expliquer le fait que les solutions issues de modèles de localisation-allocation qui appliquent la règle d'allocation la plus proche sont des solutions connexes pour le problème de sectorisation, mais pas forcément équilibrées, donc optimales pour le problème de sectorisation. Dans cet exemple, nous utilisent les modèles de p-médian et PLCE (problème de localisation avec charge équilibre) qui

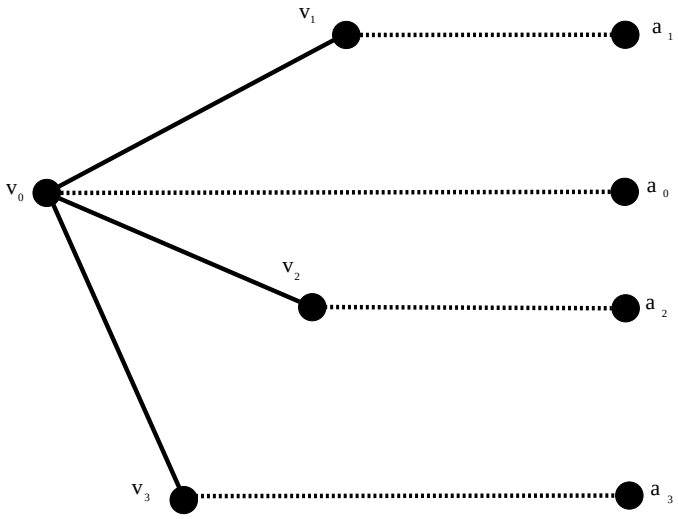

FIGURE 3.9 – Preuve de la proposition 1

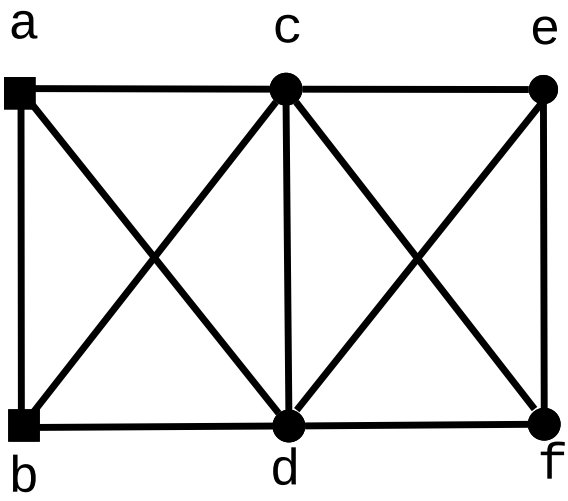

FIGURE 3.10 – Exemple appuyant la proposition 1

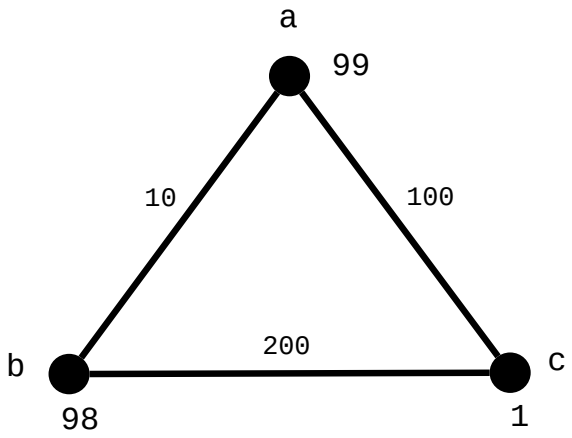

FIGURE 3.11 – Petit graphe avec 3 sommets

sont proches de notre problème de sectorisation pour trouve une solution. Dans ce graphe, nous avons trois sommets a, b et c, et les poids de sommets et les distances entre eux sont : $w_a = 99$, $w_b = 98$, $w_c = 1$; $d_{ab} = 10$, $d_{ac} = 100$, $d_{bc} = 200$. Supposons que la résolution de problèmes de localisation-allocation cherchent à localiser deux activités parmi ces trois sommets. De même, nous considérons le problème de sectorisation à deux secteurs avec des valeurs désirées $e_1 = e_2 = 99$.

Pour le problème de sectorisation dans cet exemple, si nous mettons secteur $S_1 = \{b, c\}$, secteur $S_2 = \{a\}$, la valeur obtenue de chaque secteur est : $f_1 = w_b + w_c = 99$, $f_2 = w_a = 99$. La valeur de fonction objectif de sectorisation est 0 et cette solution est la meilleure solution pour le problème de sectorisation.

Le modèle de p-médian cherche à localiser les deux sites d'activités tel que le coût total est minimal. Selon ce modèle, la solution est d'implanter les deux sites d'activités sur les sommets a et b avec un coût total de 100 : le point de demandes c sera affecté à l'activité a qui est l'activité la plus proche, et le coût total $f = w_c d_{ca} = 1 \times 100 = 100$. Donc la meilleure solution issue du modèle de p-médian correspond à la solution suivante pour le problème de sectorisation : secteur $S_1 = \{a, c\}$ et secteur $S_2 = \{b\}$. La valeur obtenue pour chaque secteur est : $f_1 = w_a + w_c = 100$, $f_2 = w_b = 98$. La valeur de fonction objectif du problème de sectorisation est $\max\{|f_1 - e_1|, |f_2 - e_2|\} = 1$.

Le modèle de PLCE essaie de trouver, dans cet exemple, deux sites d'activité de telle sorte que chaque activité a une charge équilibrée en minimisant la charge maximale de chaque activité. La solution issue de ce modèle est, ici, la même que celle du p-médian. Les sites d'activités serons localisés sur les sommets a et b, l'activité a a une charge de 100 et l'activité b a une charge de 98 : si nous choisissons a et b à localiser, le point de demandes c sera affecté à l'activité a, et la charge maximale entre chaque activité est $f = w_a + w_c = 99 + 1 = 100$. Donc les solutions issues de ces deux modèles sont les mêmes. Donc la solution optimale du modèle de PLCE peut ne pas être optimale pour le problème de sectorisation.

Nous avons développé deux heuristiques HX_1 et HX_2 pour trouver une solution au

problème de sectorisation bicritère sans pôle prédéfini. Ces heuristiques sont des algorithmes de recherche locale qui sont inspirées des travaux de [Berman *et al.*, 2009] pour le problème de PLCE.

La recherche locale est une métaheuristique utilisée pour résoudre des problèmes d'optimisation difficiles. Un algorithme de recherche locale passe d'une solution à une autre dans l'espace de recherche jusqu'à ce qu'une condition d'arrêt soit atteinte. Un algorithme de recherche locale part d'une solution candidate et la déplace de façon itérative vers une solution meilleure. Cette méthode est applicable seulement si une notion de voisinage est définie sur l'espace de recherche. Habituellement, chaque solution candidate a plus d'une solution voisine et le choix de celle vers laquelle se déplacer est pris en utilisant seulement l'information sur les solutions voisines de la solution courante. La condition d'arrêt de la recherche locale peut être une limite du temps de calcul ou quand la meilleure solution trouvée par l'algorithme ne peut plus être améliorée. Les algorithmes de recherche locale sont des algorithmes sous-optimaux puisque la recherche s'arrête souvent sur un optimum local.

Les heuristiques HX_1 et HX_2 partagent la même structure. Nous définissons d'abord la notion de voisinage un ensemble de pôles $C = \{c_1, ..., c_p\}$. Avec cet ensemble C, nous pouvons avoir un nouveau ensemble de pôles en permutant entre un élément $c_i \in C$ et un élément $v_k \in V \backslash C$. Ce nouveau ensemble de pôles $C_{ik} = C \backslash \{c_i\} \cup \{v_k\}$ est voisin de l'ensemble C. Nous expliquons les étapes principales de ces deux heuristiques. Premièrement, elles génèrent p pôles aléatoirement dans l'ensemble V pour obtenir un premier ensemble $C = \{c_1, ..., c_p\}$. A partir de ces pôles, nous trouvons une solution initiale à l'aide d'une heuristique comme celles présentées dans la section 3.1. La première heuristique que nous testons, notée HX_1, est directement inspirée du problème PLCE. Il s'agit d'un algorithme par construction progressive qui affecte itérativement les éléments géographiques en utilisant la règle d'allocation au secteur le plus proche. La distance prise en compte d'un élément v_i à un élément v_k est la distance Euclidienne entre les deux points p_i et p_k qui représentent les barycentres de ces deux éléments.

Le second heuristique, notée HX_2, utilise l'heuristique H_2 (voir Section 3.1) pour trouver une solution quand les pôles sont fixés. A partir d'un ensemble de pôles C, nous générons tous ses voisins et évaluons la solution de sectorisation correspondante. Si la solution de sectorisation ne satisfait pas l'ϵ-contrainte ou la δ-contrainte alors nous mettrons sa valeur de fonction objectif Z^1 à $+\infty$ (solution infaisable). S'il existe des solutions qui sont meilleures que la solution courante, nous mettons à jour la solution courante avec la meilleure solution et nous réitérons la recherche. Ces deux heuristiques sont décrites formellement dans les Figures 3.12 et 3.13.

Nous avons testé ces deux heuristiques sur des instances générées aléatoirement. Ces instances ont été générées presque de la même façon que les instances générées pour le problème de sectorisation bicritère à partir de pôles. Dans chaque instance, un terrain géographique est représenté par un graphe qui est généré avec l'aide de la triangulation de Delaunay. Les valeurs statistiques $[w_i^1, w_i^2]$ associées à chaque sommet v_i du graphe G sont deux entiers aléatoires entre 1 et 100. La valeur de référence du premier critère associées à chaque secteur S_j est obtenue par : $e_j^1 = \sum_{v_i \in G} w_i^1 / p$. Quand un nombre d'éléments n est fixé, nous avons quatre valeurs pour le nombre de secteurs p : $\lceil 5\% \times n \rceil$, $\lceil 10\% \times n \rceil$,

Heuristique HX_1

Step 1. Choisir p éléments aléatoirement comme pôles $C = \{c_1, ..., c_p\}$,
Utiliser la règle d'allocation au plus proche pour trouver la solution initiale S et sa valeur de fonction objectif Z^1

Step 2. $\forall c_i \in C$, $\forall v_k \in V \backslash C$,
Calculer de nouveaux ensembles de pôles $C_{ik} = C \backslash \{c_i\} \cup \{v_k\}$,
Calculer la solution correspondante S_{ik} et la valeur Z^1_{ik}.
par la règle d'allocation au plus proche.

Step 3. **Si** $min_{i,k} Z^1_{ik} \geq Z^1$, **Alors** retourner (S, Z^1) ; **Fin**.

Step 4. $(i', k') = \arg min(Z_{ik})$, $S = S_{i'k'}$, $Z^1 = Z^1_{i'k'}$, $C = C_{i'k'}$.
Aller à *Step 2.*

FIGURE 3.12 – Heuristique HX_1

Heuristique HX_2

Step 1. Choisir p éléments aléatoirement comme pôles $C = \{c_1, ..., c_p\}$,
Calculer (S, Z^1) := H_2(C)

Step 2. $\forall c_i \in C$, $\forall v_k \in V \backslash C$,
Calculer de nouveaux ensembles de pôles $C_{ik} = C \backslash \{c_i\} \cup \{v_k\}$,
Calculer (S_{ik}, Z^1_{ik}) := H_2(C).

Step 3. **Si** $min_{i,k} Z^1_{ik} \geq Z^1$, **Alors** retourner (S, Z^1) ; **Fin**.

Step 4. $(i', k') = \arg min(Z_{ik})$, $S = S_{i'k'}$, $Z^1 = Z^1_{i'k'}$, $C = C_{i'k'}$.
Aller à *Step 2.*

FIGURE 3.13 – Heuristique HX_2

Algo	n	p	$\%Infai$	Dev_{min}	Dev_{moy}	Dev_{max}	Best	T_{min}	T_{moy}	T_{max}
HX_1	50	3	0.00	400.00	2157.14	11700.00	0.00	0.23	0.29	0.41
HX_2	50	3	0.00	0.00	0.00	0.00	100.00	0.14	0.21	0.47
HX_1	50	5	0.00	155.56	1331.23	8100.00	0.00	0.52	0.83	1.20
HX_2	50	5	0.00	0.00	0.00	0.00	100.00	0.31	0.55	1.30
HX_1	50	8	0.00	100.00	659.11	2300.00	0.00	1.22	2.00	2.86
HX_2	50	8	0.00	0.00	0.00	0.00	100.00	0.51	1.11	2.14
HX_1	50	10	0.00	88.00	425.05	1387.50	0.00	0.76	2.99	5.33
HX_2	50	10	0.00	0.00	0.00	0.00	100.00	0.74	1.56	2.75
HX_1	100	5	0.00	275.00	2285.60	7500.00	0.00	3.27	4.10	5.06
HX_2	100	5	0.00	0.00	0.00	0.00	100.00	2.02	3.32	5.16
HX_1	100	10	3.33	7.81	759.29	3933.33	0.00	12.80	19.19	22.93
HX_2	100	10	0.00	0.00	0.00	0.00	100.00	5.24	14.30	32.04
HX_1	100	15	23.33	40.63	452.40	1275.00	0.00	18.31	31.16	40.76
HX_2	100	15	0.00	0.00	0.00	0.00	100.00	9.90	25.88	51.47
HX_1	100	20	63.33	79.41	259.66	684.00	0.00	14.17	30.05	43.80
HX_2	100	20	0.00	0.00	0.00	0.00	100.00	19.09	50.40	90.29
HX_1	200	10	0.00	231.58	1147.74	3400.00	0.00	92.76	102.05	108.87
HX_2	200	10	0.00	0.00	0.00	0.00	100.00	54.21	96.40	160.39
HX_1	200	20	33.33	111.11	558.66	1271.43	0.00	209.18	269.77	313.04
HX_2	200	20	0.00	0.00	0.00	0.00	100.00	92.82	390.20	610.79
HX_1	200	30	40.00	0.00	533.31	1162.50	5.56	171.33	264.40	374.85
HX_2	200	30	0.00	0.00	0.66	11.91	94.44	143.89	818.71	1448.34
HX_1	200	40	83.33	144.44	239.81	500.00	0.00	127.96	188.83	265.45
HX_2	200	40	0.00	0.00	0.00	0.00	100.00	94.78	1107.41	1946.10

TABLE 3.7 – *Comparaisons entre HX_1 et HX_2*

$\lceil 15\% \times n \rceil$ et $\lceil 20\% \times n \rceil$. Le nombre d'éléments n égale à 50, 100 et 200. Pour chaque couple de (n, p), nous générons 30 instances. Nous fixons par ailleurs : $\epsilon = 200\% \times \sum_{v_i \in G} w_i^1/p$ et $\delta = 250\% \times \sum_{v_i \in G} w_i^1/p$. Nous lançons 300 exécutions de HX_1 et 1 exécutions de HX_2 pour chaque instance.

Les critères que nous utilisons pour évaluer ces deux heuristiques sont : (1) le pourcentage d'infaisabilité $Infai$ qui évalue le pourcentage d'instances pour lesquelles une heuristique n'a pas réussi à trouver une heuristique ; (2) la déviation minimale Dev_{min}, la déviation moyenne Dev_{moy} et la déviation maximale Dev_{max}. La déviation (en pourcentage) est définie par $((Z_H^1 - Z_{Best}^1)/Z_{Best}^1) \times 100$ avec Z_{Best}^1 la meilleure valeur de la fonction objectif Z^1 trouvée parmi les deux heuristiques et Z_H^1 la valeur de fonction objectif Z^1 trouvée par l'heuristique H (HX_1 ou HX_2). Les déviations pour une heuristique H sont calculées uniquement sur les instances qu'elle résout ; (3) le pourcentage $Best$ indique le pourcentage d'instances pour lesquelles une heuristique retourne la meilleure solution ; (4) le temps de calcul en seconde. Le temps minimal T_{min}, le temps moyen T_{moy} et le temps maximal T_{max} sont calculés. Les tests expérimentaux ont été réalisés sur une machine équipée d'un processeur Pentium Core 2 Duo 3.0Ghz et de 2.48G de RAM. Nous avons utilisé le langage C++ et la librairie standard (STL) pour développer les heuristiques.

Dans le tableau 3.7, nous pouvons remarquer que HX_2 trouve plus souvent une solution faisable que HX_1. La raison est que HX_2 est moins sensible que HX_1 à l'ϵ-contrainte et la δ-contrainte. Nous remarquons aussi à partir des colonnes de déviations, la qualité des solutions trouvées par HX_2est largement meilleure que celles trouvées par HX_1. Selon la colonne du pourcentage *Best*, HX_2 a aussi beaucoup plus de chance à trouver la meilleure solution parmi ces deux heuristiques. Mais sur le temps de calcul, HX_1 est beaucoup plus rapide que HX_2, et quand la taille des instances devient plus grande, le temps de calcul de HX_2 augmente plus rapidement que celui de HX_1.

3.3 Conclusion

Dans ce chapitre, nous avons présenté des heuristiques permettant de résoudre les problèmes de sectorisation.

Pour le problème de sectorisation bicritère à partir de pôles (le problème $P1$), nous avons tout d'abord développé trois heuristiques pour trouver rapidement une solution faisable de sectorisation. Et puis nous avons présenté une formulation mathématique de ce problème de sectorisation. Ce travail a fait l'objet d'une communication à la conférence nationale *la Société Française de Recherche Opérationnelle et d'Aide à la Décision* (ROADEF2010)[Tang *et al.*, 2010a]. Ensuite, nous avons utilisé un algorithme des boîtes pour trouver une représentation du front de Pareto de ce problème. Cette méthode a fait l'objet d'une conférence internationale *MOPGP'10*[Tang *et al.*, 2010b].

Pour le problème de sectorisation bicritère sans pôle prédéfini (le problème $P2$), nous avons tout d'abord étudié les problèmes de localisation-allocation qui est proche du problème de sectorisation, et puis nous nous avons développé deux heuristiques pour résoudre notre problème. Ce travail a fait l'objet d'une communication à la conférence nationale *ROADEF2011*[Tang *et al.*, 2011c].

Pour le problème de sectorisation bicritère avec un ou plusieurs pôles candidats prédéfinis pour chaque secteur(le problème $P3$), nous avons commencé à étudier et modéliser ce problème, cependant il reste encore des tests à finir.

Chapitre 4

Prétraitement et méthode exacte

Dans le chapitre précédent, nous avons présenté des méthodes heuristiques qui trouvent rapidement une solution approchée pour les problèmes de sectorisation bicritères. Dans ce chapitre, nous nous concentrons sur le problème de sectorisation bicritère sans pôle prédéfini puisque ce problème est plus général que le problème de sectorisation bicritère à partir de pôles. Au lieu de trouver une solution approchée, nous nous intéressons à trouver une solution optimale en utilisant une procédure par séparation et évaluation (*Branch and Bound*). Comme cette méthode a une complexité exponentielle, nous étudions dans un premier temps comment réduire optimalement la taille du problème. Pour cela, nous présentons une formulation mathématique de ce problème. Et puis nous ajoutons quelques inégalités valides pour restreindre l'espace des solutions et développons une procédure de prétraitement pour réduire le nombre de variables. À la fin, nous présentons la procédure par séparation et évaluation et nous l'appliquons pour résoudre ce problème de sectorisation.

4.1 Prétraitement pour le problème de sectorisation bicritère sans pôle prédéfini

4.1.1 Une formulation mathématique

Dans la section 3.1.1, nous avons présenté une formulation mathématique du problème de sectorisation bicritère avec des pôles fixés. Nous présentons ici une formulation mathématique du problème de sectorisation bicritère sans pôle prédéfini qui ressemble à celle du problème avec des pôles. Nous rappelons brièvement la définition du problème de sectorisation bicritère sans pôle prédéfini. Soit $G = (V, E)$ un graphe non-orienté connecté avec l'ensemble des sommets $V = \{v_1, ..., v_n\}$ et l'ensemble des arêtes E. Un sommet v_i dans le graphe représente un élément sur la carte géographique. Une arête $(v_i, v_k) \in E$ relie deux sommets v_i et v_k dans le graphe G si et seulement si les éléments associés à v_i et v_k sur la carte sont voisins géographiquement. À chaque sommet $v_i \in V$, on associe un vecteur d'évaluation à deux dimensions $[w_i^1, w_i^2]$ pour représenter ses deux informations statistiques. Pour chaque paire de sommets v_i et v_k, soit a_{ik} un indice binaire qui est égale à 1 s'il existe une arête entre v_i et v_k. On connaît le nombre de secteurs p à construire.

L'objectif est d'avoir une p-partition de ces éléments en minimisant les deux fonctions objectif suivantes :

$$(i) Z^1 = \max_j |\sum_{v_i \in S_j} w_i^1 - e_j^1|,$$

$$(ii) Z^2 = \max_{j \neq j'} |\sum_{v_i \in S_j} w_i^2 - \sum_{v_{i'} \in S_{j'}} w_{i'}^2|,$$

avec $\{S_1, ..., S_j, ..., S_p\}$ qui forment une partition de V et chaque secteur S_j est connexe dans G. On connaît, de plus, e_j^1 est une valeur de référence pour le secteur S_j pour le premier critère. On a aussi une contrainte additionnelle qui impose que la valeur de chaque secteur pour le deuxième critère ne peut pas dépasser une valeur prédéfinie par le Décideur, notée δ. Nous notons cette contrainte par δ-contrainte.

Pour garantir la connexité, nous construisons un arbre couvrant correspondant à chaque secteur. Soit x_{ij} et y_{ijk} des variables binaires : x_{ij} est égal à 1 si et seulement si v_i est affecté au secteur S_j et y_{ijk} est égal à 1 si et seulement si l'arête (v_i, v_k) est choisie pour construire l'arbre couvrant associé au secteur S_j. Pour exclure des cycles dans un arbre couvrant, des variables supplémentaires u_{ij} sont utilisées. Nous désignons les ensembles des indices des sommets et des secteurs par $I = \{1, ..., n\}$ et $J = \{1, ..., p\}$ respectivement. Le problème de sectorisation bicritère sans pôle prédéfini peut être formulé comme un problème d'affectation avec des contraintes de connexité. Le modèle mathématique est le suivant :

Données

Le nombre de sommets n,
Le nombre de secteurs souhaités p,
La valeur de référence du premier critère associée au secteur S_j : e_j^1,
Les valeurs statistiques associées à chaque sommet v_i : $[w_i^1, w_i^2]$,
L'indice d'adjacence entre deux sommets a_{ik} :

$$a_{ik} = \begin{cases} 1 & \text{s'il existe une arête entre les sommets } v_i \text{ et } v_k \text{ dans le graphe } G, \\ 0 & \text{sinon.} \end{cases}$$

Les valeurs des ϵ et δ.

Variables

x_{ij}, exprime la relation entre le sommet v_i et le secteur S_j :

$$x_{ij} = \begin{cases} 1 & \text{si le sommet } v_i \text{ est affecté au secteur } S_j, \\ 0 & \text{sinon.} \end{cases}$$

$$y_{ijk} = \begin{cases} 1 & \text{si l'arête } (v_i, v_k) \text{ est sélectionnée pour construire} \\ & \text{l'arbre couvrant associé au secteur } S_j \\ 0 & \text{sinon} \end{cases}$$

u_{ij}, la profondeur du sommet v_i dans l'arbre couvrant associé au secteur S_j.

Contraintes

$$\sum_{j=1}^{p} x_{ij} = 1, \quad \forall i = 1, ..., n \tag{4.1}$$

$$y_{ijk} \leq x_{ij}, y_{ijk} \leq x_{kj}, y_{ijk} \leq a_{ik}, \quad \forall i,k = 1,...,n, \quad \forall j = 1,...,p \tag{4.2}$$

$$\sum_{k=1}^{n} y_{kji} \leq x_{ij}, \quad \forall i = 1,...,n, \quad \forall j = 1,...,p \tag{4.3}$$

$$n \cdot y_{ijk} + u_{ij} \leq u_{kj} + (n-1), \quad \forall i,k = 1,...,n, \quad \forall j = 1,...,p \tag{4.4}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{n} y_{kji} = \sum_{i=1}^{n} x_{ij} - 1, \quad \forall j = 1,...,p \tag{4.5}$$

$$u_{ij} \leq n x_{ij}, u_{ij} \geq x_{ij} \quad \forall i = 1,...,n, \quad \forall j = 1,...,p \tag{4.6}$$

$$x_{ij}, y_{ijk} \in \{0,1\}, u_{ij} \in \{1,...,n\} \quad \forall i = 1,...,n, \forall j = 1,...,p \tag{4.7}$$

$$\sum_{i=1}^{n} w_i^2 x_{ij} \leq \delta, \quad \forall j = 1,...,p \tag{4.8}$$

Fonction Objectif

$$Z^1 = \min \max_{1 \leq j \leq p} |\sum_{i=1}^{n} w_i^1 x_{ij} - e_j^1| \tag{4.9}$$

$$Z^2 = \max_{1 \leq j,j' \leq p, j \neq j'} |\sum_{i=1}^{n} w_i^2 x_{ij} - \sum_{i=1}^{n} w_i^2 x_{ij'}| \tag{4.10}$$

Les contraintes (4.1) sont des contraintes d'affectation et elles assurent que chaque élément basique est affecté à exactement un secteur. Les contraintes (4.2) déclarent que une arête (v_i, v_k) ne peut pas être dans l'arbre couvrant du secteur S_j si v_i et v_k ne sont pas affectés au secteur S_j ou si v_i et v_k ne sont pas voisins. Les contraintes (4.3) indiquent qu'il ne peut pas exister plus qu'une arête entrant pour chaque sommet dans un arbre couvrant. Les contraintes (4.4) indiquent que le nombre d'arêtes est égale au nombre de sommets moins 1 pour chaque arbre couvrant. Les contraintes (4.5) et (4.6) sont les conditions d'élimination de cycles dans un arbre : elles sont dérivées du problème du voyageur de commerce par ([Miller *et al.*, 1960]). Les contraintes (4.7) sont des contraintes d'intégralité des variables de décision. Les contraintes (4.8) sont les δ-contraintes qui imposent que la valeur de chaque secteur pour le deuxième critère ne peut pas dépasser une valeur δ. Nous avons deux critères (4.9) et (4.10) à minimiser. Et comme dans le problème de sectorisation bicritère à partir de pôles, nous considérons le critère (4.9) comme le critère principal, et nous utilisons l'approche ϵ-contrainte pour borner le deuxième critère (4.10).

Leux critères (4.9) et (4.10) sont à minimiser. Le critère (4.9) est le critère principal, et nous utilisons la ϵ-contrainte pour borner le deuxième critère (4.10).

4.1.2 Des inégalités valides

Dans cette section, nous présentons une série d'inégalités valides pour le critère Z^1 lorsque le critère Z^2 est ignorée et qui peuvent être utilisées pour restreindre l'espace des solutions du problème de sectorisation. Ces inégalités sont ajoutées au modèle mathématique pour améliorer la technique de prétraitement. Ces inégalités sont obtenues par une analyse des propriétés structurelles du problème. Supposons que nous avons une solution faisable qui est déterminée par l'heuristique HX_2. Cette heuristique présentée dans le chapitre précédent est spécifiquement conçue pour le problème de sectorisation bicritère sans pôle prédéfini. La solution de HX_2 nous donne une borne supérieure sur la valeur de Z^1 dans la solution optimale, notée UB. Avec cette borne supérieure, nous ajoutons plusieurs nouvelles contraintes à la formulation initiale du problème. Ces contraintes sont redondantes pour la formulation initiale (programme linéaire en nombres entiers, PLNE), mais elles ne sont pas redondantes pour sa relaxation continue noté PLS. Elles peuvent améliorer (augmenter dans notre cas) la valeur de la solution optimale de la PLS et accélérer la résolution du PLNE par des algorithmes exacts de type *branch-and-bound* ou *branch-and-cut*. Dans la suite, nous établissons quelques caractéristiques de la solution optimale. Comme ces inégalités valides ne sont que sur le premier critère, pour simplifier la notation, dans cette section, nous utilisons e_j au lieu de e_j^1, w_i au lieu de w_i^1, Z au lieu de Z^1. Nous notons $f_j = \sum_{v_i \in S_j} w_i$ la valeur du secteur S_j.

Lemme 4.1.1 *Dans toute solution optimale,* $\forall S_j, f_j \in [e_j - UB, e_j + UB]$.

Preuve. Soit Z^* la valeur optimale du critère Z. À partir de la fonction objectif (4.9), nous avons : $\forall j \in J$, $Z^* \geq f_j - e_j$ et $Z^* \geq e_j - f_j$. Comme $Z^* \leq UB$, on obtient $UB \geq f_j - e_j$ et $UB \geq e_j - f_j$. Ainsi, nous avons $e_j - UB \leq f_j \leq e_j + UB$, $\forall j \in J$. □

Considérons maintenant, sans perte de généralité, que les valeurs de référence e_j sont indexées dans un ordre non-décroissant. Ensuite, il peut être montré qu'il existe une solution optimale dans laquelle toutes les valeurs de f_j sont également triées dans un ordre non-décroissant.

Proposition 2 *Sous l'hypothèse* $e_j \leq e_{j+1}$, $\forall j = 1, ..., p-1$, *il existe une solution optimale à la PLNE pour laquelle* $f_j \leq f_{j+1}$, $\forall j = 1, ..., p-1$.

Preuve. Supposons que S est une solution optimale avec deux secteurs S_{j_1}, S_{j_2} $(j_1, j_2 \in J)$ de telle sorte que $e_{j_1} \leq e_{j_2}$, $f_{j_1} > f_{j_2}$. Nous construisons une nouvelle solution S' basé sur S avec juste une permutation simple entre S_{j_1} et S_{j_2} :
$S'_{j_1} = S_{j_2}$, $S'_{j_2} = S_{j_1}$ et, $\forall j \in J \backslash \{j_1, j_2\}$, $S'_j = S_j$.
Dans la suite, nous prouvons que S est dominé par S'. Les valeurs objectives de S et S' sont :
$Z_S = max(max(|f_{j_1} - e_{j_1}|, |f_{j_2} - e_{j_2}|), max_{j \in J \backslash \{j_1, j_2\}}(|f_j - e_j|))$,
et,
$Z_{S'} = max(max(|f_{j_2} - e_{j_1}|, |f_{j_1} - e_{j_2}|), max_{j \in J \backslash \{j_1, j_2\}}(|f_j - e_j|))$.
Les deuxièmes parties de Z_S et $Z_{S'}$ sont les mêmes, donc nous allons nous concentrer sur les

premières parties. Soit $Z'_S = max(|f_{j_1} - e_{j_1}|, |f_{j_2} - e_{j_2}|)$ et $Z'_{S'} = max(|f_{j_2} - e_{j_1}|, |f_{j_1} - e_{j_2}|)$. Nous divisons les relations entre e_{j_1} et f_{j_1}, e_{j_2} et f_{j_2} en 4 cas exhaustif.
Cas 1 : $e_{j_1} \geq f_{j_1}$; $e_{j_2} \geq f_{j_2}$
Ce cas implique que $e_{j_2} \geq e_{j_1} \geq f_{j_1} > f_{j_2}$. Donc, nous avons
$Z'_S = max(e_{j_1} - f_{j_1}, e_{j_2} - f_{j_2}) = e_{j_2} - f_{j_2}$, $Z'_{S'} = max(e_{j_1} - f_{j_2}, e_{j_2} - f_{j_1})$.
Puisque $e_{j_1} - f_{j_2} \leq e_{j_2} - f_{j_2}$ et $e_{j_2} - f_{j_1} < e_{j_2} - f_{j_2}$, nous avons $Z'_S \geq Z'_{S'}$.
Cas 2 : $e_{j_1} < f_{j_1}$; $e_{j_2} < f_{j_2}$
Ce cas implique que $f_{j_1} > f_{j_2} > e_{j_2} \geq e_{j_1}$. Donc, nous avons
$Z'_S = max(f_{j_1} - e_{j_1}, f_{j_2} - e_{j_2}) = f_{j_1} - e_{j_1}$, $Z'_{S'} = max(f_{j_2} - e_{j_1}, f_{j_1} - e_{j_2})$.
Puisque $f_{j_2} - e_{j_1} < f_{j_1} - e_{j_1}$ et $f_{j_1} - e_{j_2} \leq f_{j_1} - e_{j_1}$, nous avons $Z'_S \geq Z'_{S'}$.
Cas 3 : $e_{j_1} \geq f_{j_1}$; $e_{j_2} < f_{j_2}$
Ce cas implique que $f_{j_2} > e_{j_2} \geq e_{j_1} \geq f_{j_1}$ ce qui entraîne une contradiction avec l'hypothèse $f_{j_1} > f_{j_2}$. Donc, ce cas n'existe pas.
Cas 4 : $e_{j_1} < f_{j_1}$; $e_{j_2} \geq f_{j_2}$
Dans ce cas $Z'_S = max(f_{j_1} - e_{j_1}, e_{j_2} - f_{j_2})$, et nous considérons 4 sous-cas.

Cas 4.1 : $f_{j_2} \geq e_{j_1}$; $f_{j_1} \geq e_{j_2}$
Dans ce cas nous avons $Z'_{S'} = max(f_{j_2} - e_{j_1}, f_{j_1} - e_{j_2})$. Puisque $f_{j_1} - e_{j_1} > f_{j_2} - e_{j_1}$ et $f_{j_1} - e_{j_1} \geq f_{j_1} - e_{j_2}$, nous avons $Z'_S \geq Z'_{S'}$.
Cas 4.2 : $f_{j_2} \geq e_{j_1}$; $f_{j_1} < e_{j_2}$
Dans ce cas, $Z'_{S'} = max(f_{j_2} - e_{j_1}, e_{j_2} - f_{j_1})$. Puisque $f_{j_1} - e_{j_1} > f_{j_2} - e_{j_1}$ et $e_{j_2} - f_{j_2} > e_{j_2} - f_{j_1}$, nous avons $Z'_S \geq Z'_{S'}$.
Cas 4.3 : $f_{j_2} < e_{j_1}$; $f_{j_1} \geq e_{j_2}$
Dans ce cas, $Z'_{S'} = max(e_{j_1} - f_{j_2}, f_{j_1} - e_{j_2})$. Puisque $e_{j_2} - f_{j_2} \geq e_{j_1} - f_{j_2}$ et $f_{j_1} - e_{j_1} \geq f_{j_1} - e_{j_2}$, nous avons $Z'_S \geq Z'_{S'}$.
Cas 4.4 : $f_{j_2} < e_{j_1}$; $f_{j_1} < e_{j_2}$
Dans ce cas, $Z'_{S'} = max(e_{j_1} - f_{j_2}, e_{j_2} - f_{j_1})$. Puisque $e_{j_2} - f_{j_2} \geq e_{j_1} - f_{j_2}$ et $e_{j_2} - f_{j_2} > e_{j_2} - f_{j_1}$, nous avons $Z'_S \geq Z'_{S'}$.

Dans tous les cas, S est dominé par S'. Ce qui contredit le fait que S est une solution optimale. □

Nous pouvons utiliser le lemme 4.1.1 et la contrainte de connexité pour générer d'autres inégalités valides. Pour un élément basique v_i, nous notons l'ensemble de ses voisins par $M_{\{v_i\}} \subseteq V$. Nous pouvons voir que, si sa valeur statistique w_i n'est pas assez grande pour le secteur S_j ($w_i < e_j - UB$), alors il peut être dans le secteur S_j sous la condition qu'au moins un de ses voisins appartienne aussi à S_j.

Lemme 4.1.2 *$\forall i \in I$, $\forall j \in J$, si $w_i < e_j - UB$, alors nous avons : $x_{ij} \leq \sum_{k \in M_{\{v_i\}}} x_{kj}$.*

Preuve. Supposons qu'il existe une solution optimale de telle sorte qu'un élément basique v_i avec $w_i < e_j - UB$, est le seul élément dans un secteur S_j, alors on a $f_j = w_i < e_j - UB$ et cette solution ne peut pas être optimale. □

Nous donnons un petit exemple du problème de sectorisation sans pôle prédéfini, illustré dans la Figure 4.1. Considérons $V = \{a, b, c, d, e, f\}$ les six éléments basiques (sommets), avec les valeurs $w_a = 9$, $w_b = 2$, $w_c = 6$, $w_d = 7$, $w_e = 2$, $w_f = 2$ et $e_1 = 13$, $e_2 = 15$. La ligne en pointillé dans la figure créée deux secteurs : le premier secteur S_1 contient les

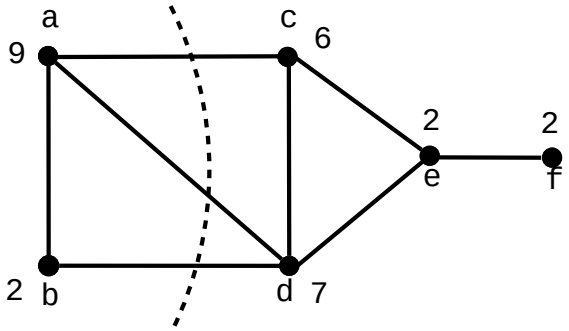

FIGURE 4.1 – Un exemple illustratif

sommets a et b, tandis que le deuxième secteur S_2 contient les sommets c, d, e et f. La valeur de chaque secteur est $f_1 = 9 + 2 = 11$ et $f_2 = 6 + 7 + 2 + 2 = 17$.

Cette solution donne une borne supérieure $UB = 2$. Par ailleurs, le sommet b est un petit élément pour le secteur S_1 car $w_b < e_1 - UB = 11$. L'ensemble de ses voisins est désigné par $M_{\{b\}} = \{a, d\}$. Le Lemme 4.1.2 indique que dans une solution optimale, le sommet b peut être dans le secteur S_1 sous la condition que le sommet a ou le sommet d sont dans le secteur S_1. Cela peut être exprimé par $x_{b1} \leq x_{a1} + x_{d1}$.

Nous pouvons étendre le Lemme 4.1.2 à un ensemble spécifique de sommets, désigné par $R \subseteq V$. Si la valeur de R, désignée par $f_R = \sum_{v_i \in R} w_i$, n'est pas assez grande pour le secteur S_j ($f_R < e_j - UB$), alors il doit exister au moins un autre élément, qui n'appartient pas à R, dans le secteur secteur S_j. Similairement, si la valeur de R est trop grande pour le secteur S_j ($f_R > e_j - UB$), alors les éléments de R ne peuvent pas être tous dans le secteur S_j.

Lemme 4.1.3 *$\forall R \subseteq V$, $\forall j \in J$, si $f_R < e_j - UB$, alors nous avons : $\sum_{v_i \notin R} x_{ij} \geq 1$.*

Lemme 4.1.4 *$\forall R \subseteq V$, $\forall j \in J$, si $f_R > e_j + UB$, alors nous avons : $\sum_{v_i \in R} x_{ij} \leq |R| - 1$.*

Les Lemmes 4.1.3 et 4.1.4 peuvent être prouvés par le même raisonnement que dans la preuve du 4.1.2. Dans le Lemme 4.1.3, on peut avoir l'intuition que, au lieu de considérer les éléments basiques $v_i \notin R$, nous pouvons nous concentrer sur les éléments basiques $v_i \in M_R$, avec M_R le voisinage de R : $M_R = \{v_i | v_i \in V$ et v_i est voisin d'un élément dans $R\}$. Mais cela n'est vrai que sous la condition que R est déjà affecté au secteur S_j. Cela peut impliquer l'ajout de variables binaires supplémentaires et puis causer, à la fin, des inégalités faibles à cause de la relaxation PLS.

Reprenons l'exemple de la Figure 4.1 pour illustrer ces deux lemmes. Pour le Lemme 4.1.3, nous pouvons trouver un ensemble de cardinalité maximale $R = \{b, c, e, f\}$ pour le secteur S_2 telle que $f_R = w_b + w_c + w_e + w_f = 12 < e_2 - UB = 13$. D'après le Lemme 4.1.3, il doit exister au moins un élément dans le complément de R affecté au secteur S_2. Cela peut être exprimé par $x_{a2} + x_{d2} \geq 1$. Pour le Lemme 4.1.4, nous pouvons trouver un ensemble de cardinalité minimale $R = \{a, d\}$ pour le secteur S_1 telle que $f_R = w_a + w_d =$

$16 > e_1 + UB = 15$. D'après le Lemme 4.1.4, ces deux éléments ne peuvent pas être affectées au secteur S_1 dans le même temps. Cela peut être exprimé par $x_{a1} + x_{d1} \leq 1$.

Dans la suite, nous introduisons le problème de l'arbre de Steiner et la notion de séparateur de sommets en théorie des graphes. Ils sont utilisés pour renforcer les inégalités valides du Lemme 4.1.4. Étant donné un graphe non-orienté $G = (V, E)$ représenté par un ensemble de n sommets V et un ensemble d'arêtes pondérées E, le problème de l'arbre de Steiner consiste à connecter un sous-ensemble de k sommets donnés R, appelés terminaux, par un arbre de coût minimal ([Hakimi, 1971]). C'est un des premiers problèmes montrés à être NP-difficile par [Karp, 1972]. Cependant, certains algorithmes avec un temps de calcul exponentiel ont été considérés comme acceptables si la taille du problème n'est pas trop grande. Une méthode exacte bien connue pour résoudre ce problème est la méthode de programmation dynamique présentée par [Dreyfus et Wagner, 1972]. Cette méthode a été améliorée après par [Niedermeier, 2008]. Nous rappelons ici cet algorithme de programmation dynamique. Soit $X \subseteq R$, et $v \in V \backslash X$. Soit $F(X \cup \{v\})$ le poids d'un arbre de Steiner minimal reliant tous les sommets de $X \cup \{v\}$ dans le graphe G et soit $d(u, v)$ la valeur du plus court chemin, en termes de poids, entre les sommets u et v. Soit $X \neq \emptyset$, $X \subseteq R$, et $v \in V \backslash X$. alors,

$F(X \cup \{v\}) = min\{min_{u \in X}\{F(X) + d(u, v)\}, min_{u \in V \backslash X}\{F_u(X \cup \{u\}) + d(u, v)\}\}$ avec
$F_u(X \cup \{u\}) = min_{X' \neq \emptyset, X' \subset X}\{F(X' \cup \{u\}) + F((X \backslash X') \cup \{u\})\}$.

Cet algorithme nécessite un temps proportionnel à $O(k^3 n + k^2 n^2 + n^2 logn + nm)$. Notons que la distance $d(u, v)$ peut être calculée par un algorithme du plus court chemin en prenant les poids des arêtes comme des distances.

La version orientée du problème de l'arbre de Steiner est une extension de la version non-orienté. Selon [Charikar *et al.*, 1999], la version orientée du problème de l'arbre de Steiner est définie comme suit. Étant donné un graphe orienté $G = (V, A)$, une racine spécifiée $r \in V$, et un ensemble de terminaux $R \subseteq V$ ($|R| = k$), l'objectif est de trouver l'arborescence de coût minimal de la racine r et couvrant tous les sommets de R. L'algorithme de programmation dynamique de Niedermeier peut être facilement adapté pour résoudre la version orientée du problème de l'arbre de Steiner. Ceci est fait en prenant en compte dans le calcul de $d(u, v)$ le fait que l'on a des arcs et non des arêtes. A l'initialisation de l'algorithme de programmation dynamique, on pose $X = \{r\}$.

Dans le problème de l'arbre de Steiner, les poids sont affectés sur des arêtes, mais dans le problème de sectorisation, les poids sont affectés sur des sommets. Pour utiliser l'algorithme de Niedermeier pour résoudre le problème de l'arbre de Steiner dans un graphe pondéré sur les sommets, il est nécessaire de transformer ce graphe "sommet-pondéré" (le graphe original G) en un graphe "arc-pondéré" (le graphe transformé G') : pour chaque sommet v_i dans G, nous créons deux sommets correspondants v_i^I (le sommet qui reçoit des arcs) et v_i^O (le sommet qui émet des arcs) dans G'. Nous construisons un arc de v_i^I à v_i^O avec un poids de w_i (le poids de v_i dans G). Pour chaque paire de sommets dans G, v_i et v_k, s'il existe une arête de v_i à v_k, nous construisons un arc de v_i^O à v_k^I avec un poids de 0 dans G'. Cette procédure de transformation est illustrée dans la Figure 4.2. Il est facile de montrer que le graphe obtenu G' est fortement connecté. Maintenant, nous illustrons pourquoi une arborescence de Steiner minimale de la racine r dans G' peut être associée

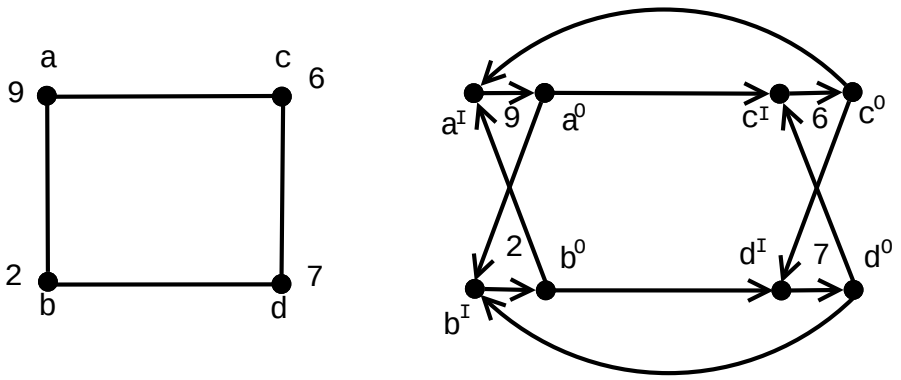

FIGURE 4.2 – Un petit exemple de transformation de graphe

à un arbre de Steiner minimal dans G. Tout d'abord, si R est l'ensemble de terminaux dans G, alors l'ensemble de terminaux R' dans G' comprend à tous les sommets v_i^O et v_i^I de G' associés aux sommets $v_i \in R$. La racine r est choisie dans R' et elle correspond à un sommet qui reçoit des arcs, c'est à dire, de type x^I. Il est clair que pour chaque arborescence ST de la racine r et couvrant tous les sommets dans R', les arcs de type (x^I, x^O) sont dans ST (puisque par la construction x^O est le seul successeur direct de x^I dans G'). Par conséquence, une arborescence de coût minimal de la racine r contient tous les arcs avec des poids associés aux sommets dans R, cela est vrai quel que soit $r \in R'$ de type x^I. Par ailleurs, tous les arcs de poids 0 dans l'arborescence correspondent à une arête dans G. Ainsi, une arborescence de coût minimal calculé ci-dessus peut être facilement transformé en un arbre de Steiner minimal avec des poids sur les sommets.

Nous pouvons renforcer le Lemme 4.1.4 en résolvant des problèmes de calcul d'arbres de Steiner. Pour un ensemble de sommets (terminaux) R, si le coût de l'arbre de Steiner correspondant, noté par t_R, est trop grand pour le secteur S_j ($t_R > e_j + UB$), alors les sommets de R ne peuvent pas être tous dans le secteur S_j.

Proposition 3 *$\forall R \subseteq V$, $\forall j \in J$, si $t_R > e_j + UB$, alors nous avons : $\sum_{v_i \in R} x_{ij} \leq |R| - 1$.*

Cette proposition domine le Lemme 4.1.4 puisque pour tout ensemble de sommets $R \subseteq V$, nous avons toujours $f_R \leq t_R$. Par exemple, dans la Figure 4.1, nous avons un ensemble de terminaux $R = \{a, f\}$. L'arbre de Steiner correspondant est $a \to c \to e \to f$ avec $t_R = 9 + 6 + 2 + 2 = 19 > e_1 + UB$. Avec la Proposition 3, a et f ne peuvent pas être affectés simultanément au secteur S_1 et cela peut être exprimé par $x_{a1} + x_{f1} \leq 1$. Mais nous pouvons noter que cette inégalité ne peut pas être dérivée du Lemme 4.1.4.

Un séparateur de sommets est défini comme un ensemble de sommets $Sep \subseteq V$, dont le retrait divise le graphe en K ($K \geq 2$) composants déconnectés, notés par C_k ($k \in \{1, ..., K\}$). Par exemple, dans la Figure 4.1, $\{e\}$ est un séparateur de sommets puisque sans ce sommet le graphe est divisé en 2 composants ($C_1 = \{f\}$; $C_2 = \{a, b, c, d\}$). L'ensemble des sommets $\{c, d\}$ est également un séparateur de sommets dont la cardinalité est 2.

Proposition 4 *Pour tout composant C_k ($k \in \{1,...,K\}$) généré par un séparateur de sommets donné, $Sep \subseteq V$, et $\forall j \in J$, si $f_{C_k} < e_j - UB$ alors, $\forall v_i \in C_k$, nous avons :* $x_{ij} \leq \sum_{v_{i'} \in Sep} x_{i'j}$.

Preuve. Supposons qu'un séparateur de sommets Sep et les composants correspondants C_k, $\forall k \in \{1,...,K\}$ sont donnés. Par ailleurs, soit $j \in J$ un indice d'un secteur tel que, $\exists k$, $f_{C_k} < e_j - UB$. $\forall k$, C_k forme un sous-graphe connexe de G et les sommets de Sep sont les seuls sommets de V connectés à C_k. Par conséquence, similairement au Lemme 4.1.2, si $v_i \in C_k$ est affecté au secteur S_j, alors au moins un sommet $v_i \in Sep$ doit être affecté au secteur S_j. □

Par exemple, dans la Figure 4.1, $Sep = \{e\}$ est un séparateur. Le composant composé par $\{f\}$ est petit pour le secteur S_1 ($w_f < e_1 - UB$). Par la Proposition 4, nous avons que le sommet f peut être affecté à S_1 sous la condition que le sommet e soit affecté à S_1 : cela peut être exprimé par $x_{f1} \leq x_{e1}$.

4.1.3 La phase de prétraitement

4.1.3.1 Techniques de fixation de variables

Étant donné une formulation en PLNE d'un problème d'optimisation, le prétraitement est une opération qui peut être effectuée pour améliorer ou simplifier la formulation. Une phase de prétraitement peut conduire à la fixation des valeurs de certaines variables, à l'amélioration des bornes sur des variables, à l'identification de l'infaisabilité ou à la génération de nouvelles inégalités valides. Il s'agit d'une phase importante située entre la formulation et la résolution.

Certaines techniques de prétraitement sont décrits par [Savelsberg, 1994] et [Nemhauser et Wolsey, 1999] Dans cette thèse, nous nous concentrons sur la fixation des valeurs de variables en appliquant une technique de fixation de variables uniquement sur les variables binaires d'affectation x_{ij}. Ces variables sont plus importantes que y_{ijk} et u_{ij} pour la résolution du problème de sectorisation. Des techniques de fixation de variables sur des variables booléennes ont déjà été appliquées avec succès sur des problèmes NP-difficile d'optimisation combinatoire tels que le problème de sac-à-dos multidimensionnel ([Osorio *et al.*, 2002]) et des problèmes d'ordonnancement ([T'kindt *et al.*, 2007], [Baptiste *et al.*, 2010]). Ces techniques exploitent des informations de la relaxation linéaire tels que les coûts réduits des variables hors base ou même les pseudo-coûts pour les variables de base.

Nous expliquons brièvement ici cette technique de fixation. Considérons un problème (P) définit de la façon suivante

$$\min \sum_{i=1}^{n} z = c_i x_i \tag{4.11}$$

sous contraintes

$$Ax = b \tag{4.12}$$

$$x_i \in \{0,1\} \quad \forall i = 1,...,n. \tag{4.13}$$

La matrice A a m lignes (autant qu'il y a de contraintes) et n colonnes, avec $n > m$. Nous notons (LP) sa relaxation continue sous la forme standard :

$$\min \sum_{i=1}^{n} z = c_i x_i \tag{4.14}$$

sous contraintes

$$Ax = b \tag{4.15}$$

$$0 \leq x_i \leq 1 \quad \forall i = 1, ..., n. \tag{4.16}$$

Si nous sélectionnons m variables de base et si nous annulons les $(n - m)$ variables hors base, la matrice A peut s'écrit comme $A = (B, N)$ où B est une matrice de base et N est une matrice hors base. De même, on peut écrire le vecteur des variables x comme $x = (x_B, x_N)$ où x_B est le vecteur des variables de base et x_N est le vecteur des variables hors base. En annulant toutes les variables hors base, le système se réduit à $Bx_B = b$, ce qui donne la solution $x_B = B^{-1}b$ et $x_N = 0$. Si l'expression $B^{-1}b$ est non négative, nous avons une solution admissible.

Le problème (LP) s'écrit aussi sous la forme suivante, que nous appelons "forme canonique utilisable associée au programme de base" :

$$x_B = B^{-1}b - B^{-1}Nx_N \tag{4.17}$$

$$z = C_B B^{-1}b + (C_N - C_B B^{-1}N)x_N \tag{4.18}$$

Nous appelons "coût réduit" de la variable hors base x_j, $j \in N$, le coefficient correspondant de chaque ligne j de l'expression $C_N - C_B B^{-1}N$, noté r_j.

Nous utilisons la solution optimale de (LP) pour reformuler le problème (P). Nous notons :
(1) z_{LP}, la solution optimale du problème (LP).
(2) r_j, le coût réduit de la variable x_j avec $r_j = 0$ si $j \in B$ et $r_j \geq 0$ si $j \in N$. B est la base associée à la solution optimale du (LP) considérée.
(3) v_j, la valeur de variable x_j dans la solution optimale de (LP) avec $x_j = 0$ si $j \in N$.
(4) a_{ij}, le coefficient modifié liant les variables de base et les variables hors base dans le tableau du simplex.
Le problème (P) peut alors se reformuler par :

$$\min z_{LP} + \sum_{j \in N} r_j x_j \tag{4.19}$$

sous contraintes

$$x_j = v_j + \sum_{i \in N} a_{ji} x_i \quad \forall j \in B \tag{4.20}$$

$$x_j = 0 \quad \forall j \in N \tag{4.21}$$

Les pseudo-coûts sont introduits dans le contexte d'une méthode énumérative de type *Branch and Bound*. Dans cette méthode, il existe des variables fractionnelles dans la solution de (LP), et nous devons choisir une variable pour définir la division. Puisque l'efficacité

de cette méthode dépend fortement de la rapidité de l'augmentation de la borne inférieure, nous voulons brancher sur une variable qui va, le plus, améliorer la borne inférieure quand cette variable est fixée à une valeur entière. Pour mesurer la quantité d'augmentation par unité sur la fonction objectif si x_j est fixé à 0 ou 1, des pseudo-coûts P_j^- et P_j^+ sont associés. Pour une variable fractionnelle x_j, avec $j \in B$, [Driebeek, 1966] propose des pénalités L_j et U_j permettant de calculer une borne inférieure à l'augmentation de z_{LP} si x_j est fixé à 0 (L_j) où à 1 (U_j) :

$$P_j^- = \min_{i \in N, a_{ji} > 0} \frac{r_i}{a_{ji}} \quad \text{et} \quad P_j^+ = \min_{i \in N, a_{ji} < 0} \frac{r_i}{-a_{ji}} \tag{4.22}$$

$$\text{avec} \quad L_j = x_j^{LP} \times P_j^- \quad \text{et} \quad U_j = (1 - x_j^{LP}) \times P_j^+ \tag{4.23}$$

x_j^{LP} est la valeur de x_j de (LP). Par la suite, [Tomlin, 1971] et [Armstrong, 1974] améliorent la façon de calculer les pénalités L_j et U_j.

Nous pouvons utiliser les coûts réduits et les pseudo-coûts pour fixer les variables. D'après (4.19), nous avons : $z_{Mip} = z_{LP} + \sum_{j \in N} r_j x_j$, z_{Mip} est la solution optimale du problème (P). Soit UB et LB la borne supérieure et la borne inférieure de la solution optimale du problème (P). Donc nous avons : $UB \geq z_{LP} + \sum_{j \in N} r_j x_j$. Ainsi, pour une variable hors base x_j, $j \in N$, si son coût réduit $r_j > UB - z_{LP}$, alors nous avons nécessairement $x_j = 0$ dans une solution optimale du problème (P). Nous pouvons raisonner de même sur les variables d'écarts, notées s_j, associées aux x_j dans le problème (LP) avec $x_j + s_j = 1$, pour déduire si x_j sont à fixer à 1 dans une solution optimale du problème (P). Ainsi, pour une variable de base x_j, $j \in B$, nous pouvons utiliser les pseudo-coûts pour fixer les variables. Si $L_j > UB - LB$, alors nous avons nécessairement $x_j = 1$. De même, si $U_j > UB - LB$, alors nous avons nécessairement $x_j = 0$.

Cependant, pour notre problème de sectorisation, nous avons pu constater expérimentalement que les coûts réduits des variables hors base (et par conséquence, cela est également vrai pour les pseudo-coûts des variables de base) sont presque tous égaux à 0. Donc nous ne pouvons pas utiliser ces coûts pour fixer les variables. Par conséquence, la technique de fixation de variable utilisée dans cette thèse est entièrement basée sur la résolution itérative de problèmes de PLS dans lesquels les variables sont testées à 0 ou 1. Ce technique est plus général que l'utilisation des coûts réduits et des pseudo-coûts, mais pour notre problème, il est plus efficace, et aussi plus lent. Donc, l'idée consiste à prendre des décisions optimales par rapport à x_{ij} dans une solution optimale entière. L'objectif est alors de réduire en temps polynomial la taille de l'instance à résoudre par un algorithme en temps exponentiel comme une procédure par séparation et évaluation.

4.1.3.2 Un algorithme de prétraitement général

Soit LB une borne inférieure sur la valeur de la solution optimale de la PLNE. Elle est calculée en résolvant la relaxation en PLS du modèle de PLNE donné dans la section 4.1.1. Maintenant, considérons une variable d'affectation x_{ij}. Nous ajoutons une contrainte $x_{ij} = 1$ à la formulation de PLS et la résolvons à nouveau. Si la valeur optimale $\overline{Z}$ de ce problème relaxé est plus grand qu'une borne supérieure UB connue sur la solution du modèle de PLNE ou s'il n'y a pas de solution faisable (ce cas peut également être considéré

comme la solution a une valeur infinie), alors la variable x_{ij} est nécessairement à 0 dans une solution optimale entière. Nous pouvons utiliser le même raisonnement pour fixer x_{ij} à 1.

Pour compléter cette technique, nous utilisons la même technique en considérant des couples de variables. Soient $x_{i_1j_1}, x_{i_2j_2}$ deux variables, et deux sous-problèmes LP_1 et LP_2 : nous ajoutons deux contraintes $x_{i_1j_1} = 1$, $x_{i_2j_2} = 1$ à la relaxation linéaire pour obtenir LP_1. Similairement, deux contraintes $x_{i_1j_1} = 1$, $x_{i_2j_2} = 0$ sont ajoutées pour obtenir LP_2. Si les valeurs optimales de LP_1 et LP_2 sont toutes plus grandes que UB, alors la variable $x_{i_1j_1}$ doit être mise à 0. Un raisonnement similaire peut être dérivée pour fixer la variable $x_{i_1j_1}$ à 1.

Puisque le nombre de variables x_{ij} est $n \times p$, qui est relativement petit, nous testons d'abord cette technique de fixation sur chaque variable x_{ij}. Puis nous sélectionnons deux éléments basiques spécifiques i_1 et i_2 par certaines règles pour appliquer la second technique : nous sélectionnons deux éléments i_1 et i_2 avec des poids relativement élevés w_{i_1}, $w_{i_2} \geq \sum_{i \in I} w_i/n$, et i_1 et i_2 sont relativement loin de l'autre sur la carte avec le plus court chemin $d_{i_1,i_2} > 3.0 \sum_{i \in I} w_i/n$. Cette technique généralise l'utilisation des coûts réduits et des pseudo-coûts. L'algorithme de prétraitement pour la fixation des variables est décrit dans la Figure 4.3.

4.1.4 Renforcement du prétraitement

Dans la section 4.1.2, nous avons présenté quelques inégalités valides qui peuvent être utilisées pour restreindre l'espace des solutions du problème de sectorisation. Dans cette section, nous montrons comment générer les inégalités utilisées par les Lemmes 4.1.3 et 4.1.4, et quelques détails concernant les Propositions 3 et 4.

Pour générer les inégalités dans les Lemmes 4.1.3 et 4.1.4, il est nécessaire de construire l'ensemble R avec une cardinalité maximale pour le Lemme 4.1.3 et une cardinalité minimale pour le Lemme 4.1.4. Prenons le cas de figure du Lemme 4.1.3 et le problème de la génération d'ensembles R maximaux : il faut générer des ensembles R d'éléments qui satisfont l'inégalité suivante : $\sum_{i \in R} w_i < e_j - UB$, avec $j \in J$. Générer ces ensembles est équivalent à générer toutes les 1-coupes pour le problème de sac-à-dos ([Osorio *et al.*, 2002]) et peut être réalisé en $O(n^2)$ pour j fixé. Ainsi, pour un secteur S_j fixé nous appliquons l'algorithme de la Figure 4.4 pour générer tous les ensemble R de cardinalité maximale.

Concernant le Lemme 4.1.4 et la génération d'ensemble R minimaux on doit satisfaire l'inégalité suivante : $\sum_{i \in R} w_i > e_j + UB$, avec $j \in J$. L'algorithme de la Figure 4.4 peut être alors facilement modifié pour générer en $O(n^2)$ tous les ensembles R de cardinalité minimale.

Dans la Proposition 3, pour les inégalités basées sur l'arbre de Steiner, nous générons tous les ensembles de terminaux R telle que $|R| \leq 3$ et la condition de la Proposition 3 soit satisfaite. Lorsque la cardinalité de l'ensemble de terminaux devient plus grande, le temps requis pour le calcul de l'arbre de Steiner augmente rapidement par rapport à la réduction du temps nécessité pour résoudre le problème de sectorisation optimalement.

Dans la Proposition 4, nous devons déterminer si un ensemble de sommets *Sep* est un séparateur ou pas. Par la définition du séparateur de sommets, si la suppression des

Algorithme de prétraitement

Step 1. $\forall i \in I, \forall j \in J$, **Si** x_{ij} n'est pas fixé, **Alors**

1.1. Mettre $x_{ij} = 0$ et résoudre la PLS : S_0 est la solution correspondante et Z_0 est la valeur de la fonction objectif.
Si $Z_0 > UB$, **Alors** x_{ij} est fixé à 1 ;
Maj UB :
Si S_0 est une solution faisable et $Z_0 < UB$, **Alors** $UB = Z_0$;

1.2. Mettre $x_{ij} = 1$ et résoudre la PLS, S_1 est la solution correspondante et Z_1 est la valeur de la fonction objectif.
Si $Z_1 > UB$, **Alors** x_{ij} est fixé à 0 ;
Maj UB :
Si S_1 est une solution faisable et $Z_1 < UB$, **Alors** $UB = Z_1$;

Step 2. Pour les couples sélectionnés $(i_1, i_2) \in I, \forall j_1 \in J$,
Si $x_{i_1 j_1}$ n'est pas fixé, **Alors**

2.1. Mettre $x_{i_1 j_1} = 0, \forall j_2 \in J$,

2.1.1. Mettre $x_{i_2 j_2} = 0$ et résoudre la PLS : S_0 est la solution correspondante et Z_0 est la valeur de la fonction objectif.
Maj UB : **Si** S_0 est une solution faisable
et $Z_0 < UB$, **Alors** $UB = Z_0$;

2.1.2. Mettre $x_{i_2 j_2} = 1$ et résoudre la PLS : S_1 est la solution correspondante et Z_1 est la valeur de la fonction objectif.
Maj UB : **Si** S_1 est une solution faisable
et $Z_1 < UB$, **Alors** $UB = Z_1$;

2.1.3. Si $\min\{Z_0, Z_1\} > UB$, alors $x_{i_1 j_1}$ est fixé à 1 ;

2.2. Mettre $x_{i_1 j_1} = 1, \forall j_2 \in J$,

2.2.1. Mettre $x_{i_2 j_2} = 0$ et résoudre la PLS : S'_0 est la solution correspondante et Z'_0 est la valeur de la fonction objectif.
Maj UB : **Si** S'_0 est une solution faisable
et $Z'_0 < UB$, **Alors** $UB = Z'_0$;

2.2.2. Mettre $x_{i_2 j_2} = 1$ et résoudre la PLS : S'_1 est la solution correspondante et Z'_1 est la valeur de la fonction objectif.
Maj UB : **Si** S'_1 est une solution faisable
et $Z'_1 < UB$, **Alors** $UB = Z'_1$;

2.2.3. Si $\min\{Z'_0, Z'_1\} > UB$, alors $x_{i_1 j_1}$ est fixé à 0.

FIGURE 4.3 – Algorithme de prétraitement

Algorithme de génération de l'ensemble avec une cardinalité maximale

Step 1. v_i sont indexés dans un ordre non-décroissant en w_i,
c'est à dire $0 \leq w_1 \leq w_2 \leq ... \leq w_n$.

Step 2. $t := 0$; $s := 0$; $m := 1$;
Tant Que $s + w_t < e_j - UB$, **Répéter**
$s := s + w_t$; $t := t + 1$;
Fin Tant Que.

Step 3. $R_0 := \{v_1, ..., v_t\}$.

Step 4. **Pour** $k := t$ **à** 1 **Faire**
Pour $l := t + 1$ **à** n **Faire**
Si $s - w_k + w_l < e_j - UB$, **Alors**
$R_m := R_0 \backslash \{v_k\} \cup \{v_l\}$; $m := m + 1$;
Fin Pour ;
Fin Pour ;

FIGURE 4.4 – Algorithme de génération de l'ensemble avec une cardinalité maximale

sommets *Sep* crée un graphe non connexe, alors *Sep* est un séparateur. Pour vérifier la connexité d'un graphe, nous utilisons l'algorithme de parcours en profondeur (ou DFS, pour *Depth First Search*) qui est la base de l'algorithme de [Tarjan, 1972]. L'algorithme DFS est un algorithme de parcours de graphe qui se décrit naturellement de manière récursive. Il explore en fait à fond les chemins un par un : pour chaque sommet, il marque le sommet actuel, et il prend le premier sommet voisin jusqu'à ce qu'un sommet n'ait plus de voisins, et revient alors au sommet père. L'algorithme DFS est un algorithme très efficace qui nécessite un temps $O(|V| + |E|)$ pour vérifier la connexité. Dans la phase de calcul, nous décidons de générer l'ensemble des séparateurs *Sep* telle que $|Sep| \leq 3$. Nous limitons à cette taille car lors que la taille du séparateur devient plus grande, les inégalités deviennent moins efficace.

4.1.5 Les résultats expérimentaux

Cette section présente les résultats expérimentaux réalisés pour évaluer l'efficacité de l'algorithme de prétraitement et les inégalités valides introduites dans les sections précédentes. Les instances sont générées aléatoirement en fonction de différentes configurations de cartes. Chaque instance est définie par un graphe planaire qui est généré à partir d'un ensemble de points donnés (les éléments basiques) dans le plan. Quatre types différents de cartes sont générés : carré, rectangle, peu perturbé et perturbé. Une carte carrée est un carré de $m \times m$ éléments basiques avec m donné. Une carte rectangulaire est un rectangle de m lignes et m' colonnes ($m \neq m'$), avec $m \times m'$ éléments basiques. Une carte peu perturbée est composée par des éléments basiques qui ont des localisation diverses. Et dans une carte perturbée des éléments basiques ont des localisation plus diverses qu'une carte peu perturbée.

Nous avons quelques paramètres pour contrôler la génération de chaque type de cartes : la largeur minimale w_{min}, la largeur maximale w_{max}, la longueur minimale l_{min}, la longueur maximale l_{max}, le pourcentage pl qui contrôle la possibilité d'un élément basique à être

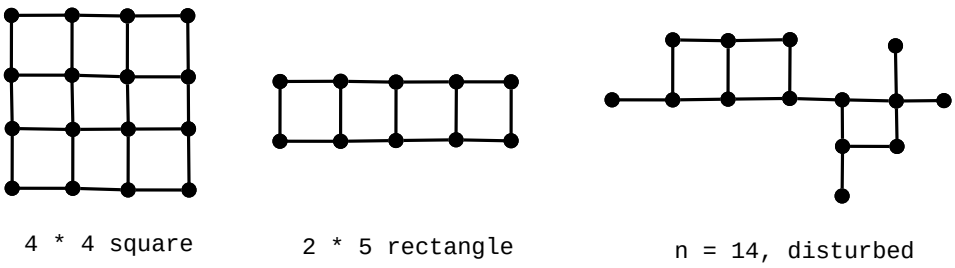

FIGURE 4.5 – Des exemples de différents types de carte

généré à un point donné sur la carte.

Pour une carte carrée, nous choisissons au hasard $m \in \{4, 5\}$, $w_{min} = w_{max} = l_{min} = l_{max} = m$, et $pl = 100\%$.

Pour une carte rectangulaire, nous choisissons au hasard $m \in \{2, 3\}$ et $m' \in \{6, 7, 8, 9\}$, $w_{min} = w_{max} = m$, $l_{min} = l_{max} = m'$, et $pl = 100\%$.

Pour une carte peu perturbée, nous choisissons au hasard $w_{min} \in \{3, ..., 7\}$ et $l_{min} \in \{3, ..., 7\}$, $w_{max} = 1.5w_{min}$, $l_{max} = 1.5l_{min}$, et $pl = 50\%$.

Pour une carte perturbée, nous choisissons au hasard $w_{min} \in \{1, 2\}$, $w_{max} = 3w_{min}$, $l_{min} = 6w_{min}$, $l_{max} = 10w_{min}$ et $pl = 30\%$.

Pour tous les types de carte, nous limitons le nombre d'éléments basiques n entre 12 et 27, et le nombre de secteurs $p \in \{2, 3, 4\}$. Par exemple, dans la Figure 4.5, il y a une carte carrée, une carte rectangulaire et une carte perturbée.

Les valeurs statistiques $[w_i^1, w_i^2]$ associées à chaque sommet v_i du graphe G sont toujours deux entiers aléatoires entre 1 et 100. La valeur de référence pour le premier critère associée à chaque secteur S_j est donnée par : $e_j^1 = \sum_{v_i \in G} w_i^1/p$. Pour chaque type de cartes et chaque valeur du nombre de secteurs p, nous générons 30 instances. Les tests expérimentaux ont été réalisés sur une machine équipée d'un processeur Pentium Core 2 Duo 3.0Ghz et de 2.48G de RAM. Toutes les formulations mathématiques sont résolues en utilisant le solveur CPLEX 12.2.

Nous effectuons deux types d'expérimentation. Tout d'abord nous ne considérons que la procédure de prétraitement et nous évaluons l'impact des inégalités valides sur le pourcentage de variables fixées. Dans un second temps nous évaluons l'impact de prétraitement sur la résolution exacte du modèle de PLNE par Cplex. Dans ce cas, d'un point de vue pratique, la relaxation linéaire utilisée par le prétraitement est convertie, après prétraitement, en un modèle de PLNE en changeant simplement le type des variables, non fixées par le prétraitement, de réel à booléen. Ainsi, la PLNE obtenue comprend toutes les inégalités valides et quelques variables fixées. Dans le solveur CPLEX, pour la résolution du PLNE, nous fixons des limites sur la mémoire utilisée par l'arbre de recherche à 650 Mo et sur le temps de calcul à 1800 secondes. CPLEX se termine si le problème est résolu à l'optimalité ou l'une de ces limites est atteinte. Dans la suite, nous testons dans un premier temps, le prétraitement sur le problème mono-critère, c'est-à-dire que nous mettons simplement les valeurs de ϵ et δ à une valeur très grande pour ignorer les ϵ-contraintes et les δ-contraintes.

TABLE 4.1 – Comparaison du pré-traitement sans et avec les inégalités valides en mono-critère

Type	n_{min}	n_{max}	p	pv_{min}	pv_{avg}	pv_{max}	t_{min}	t_{avg}	t_{max}
carré	16	25	2	0.0	0.0	0.0	0.1	0.2	0.4
				0.0	10.0	100.0	0.0	0.7	1.6
			3	0.0	0.0	0.0	0.3	0.7	1.3
				0.0	0.0	0.0	2.1	7.8	22.1
			4	0.0	0.0	0.0	0.5	1.3	2.5
				0.0	0.0	0.0	5.2	32.0	103.4
rectangle	12	27	2	0.0	0.0	0.0	0.0	0.2	0.5
				0.0	13.3	100.0	0.0	1.7	5.5
			3	0.0	0.0	0.0	0.1	0.7	1.9
				0.0	3.3	100.0	0.6	11.8	43.7
			4	0.0	0.0	0.0	0.3	1.5	4.4
				0.0	6.7	100.0	1.5	45.9	211.4
peu perturbé	12	26	2	0.0	0.0	0.0	0.0	0.2	0.6
				0.0	26.7	100.0	0.0	1.8	6.6
			3	0.0	0.0	0.0	0.1	0.8	2.0
				0.0	13.3	100.0	0.7	12.4	52.1
			4	0.0	0.0	0.0	0.3	1.7	4.0
				0.0	13.3	100.0	2.2	57.7	190.3
perturbé	12	27	2	0.0	0.0	0.0	0.1	0.3	0.6
				100.0	100.0	100.0	0.0	0.2	0.8
			3	0.0	0.0	0.0	0.2	1.0	1.9
				0.0	76.7	100.0	0.5	9.2	40.4
			4	0.0	0.0	0.0	0.3	2.2	5.2
				0.0	31.3	100.0	1.5	52.4	310.4

Dans un second temps, nous testons sur le problème en bicritère.

Considérons l'évaluation du prétraitement pour le problème mono-critère. Dans le Tableau 4.1, nous présentons les résultats obtenus par l'algorithme de prétraitement sans et avec les inégalités valides ajoutées au modèle. Pour chaque type de cartes et chaque nombre de secteurs p, il y a deux lignes de résultats : la première ligne présente les résultats du prétraitement sans inégalités valides tandis que la deuxième ligne présente les résultats du prétraitement avec les inégalités valides. Ce tableau présente le nombre minimale (n_{min}) et le nombre maximal (n_{max}) d'éléments basiques, le pourcentage minimal (pv_{min}), moyen (pv_{avg}) et maximal (pv_{max}) de variables x_{ij} fixées, et le temps CPU minimal (t_{min}), moyen (t_{avg}) et maximal (t_{max}) en secondes. Les résultats présentés dans ce tableau montrent que les inégalités valides introduites dans la Section 4.1.2 sont utiles pour fixer des variables : sans elles, aucune variable n'est fixée par le prétraitement. Par ailleurs, en termes de difficulté, les cartes carrées sont les plus difficiles à résoudre suivi des cartes rectangulaires. Plus les cartes générées sont perturbées et plus le prétraitement est efficace. Nous pouvons aussi remarquer que, sauf sur les cartes carrées, le prétraitement avec les inégalités valides est capable de fixer des variables sur toutes les configurations de cartes : le pourcentage moyen de variables fixées varie de $3,3\%$ pour les cartes rectangulaires avec 3 secteurs jusqu'à 100% pour les cartes perturbées avec 2 secteurs. Si nous ne comptons pas les cartes carrées, le pourcentage moyen de variables fixées est d'environ 32%. Les cartes carrées sont les plus difficiles à résoudre car, dans un certain sens, "tout les éléments sont équivalents" et aucune particularité structurelle ne peut être exploitée pour en déduire des affectations/interdiction d'éléments dans des secteurs.

Par ailleurs, quand p augmente, le problème devient plus difficile à résoudre. En termes de temps de calcul, le prétraitement avec les inégalités valides prend plus de temps parce que nous avons besoin de générer ces inégalités valides, et certaines d'entre elles nécessitent un temps de CPU important. C'est le cas des inégalités valides générées par la résolutions de problèmes de calcul d'arbres de Steiner (Proposition 3).

Les Tableaux 4.2 et 4.3 évaluent l'impact du prétraitement avec les inégalités valides sur la résolution de la PLNE. Le Tableau 4.2 est arrangé de la même manière que le Tableau 4.1 : pour chaque type de cartes et chaque p, la première ligne présente les résultats correspondant au modèle d'origine de la PLNE tandis que la deuxième ligne correspond à la solution de la PLNE avec un prétraitement initiale. Le Tableau 4.2 présente le pourcentage des instances résolues à l'optimalité (pc) dans les limites imposées, le nombre de nœuds minimal (nd_{min}), moyen (nd_{avg}) et maximal (nd_{max}) dans l'arbre exploré par l'algorithme de branch-and-cut de CPLEX, et le temps CPU minimal (t_{min}), moyen (t_{avg}) et maximal (t_{max}) en secondes, nécessaire pour résoudre les problèmes à l'optimalité. Notons que le temps CPU de la PLNE plus le prétraitement comprend le temps requis par le prétraitement (voir Tableau 4.1). Les résultats présentés sont calculés sur toutes les instances, y compris celles pour lesquelles une solution optimale ne peut pas être trouvée dans les limites imposées. Dans le Tableau 4.3, les mêmes résultats que dans le Tableau 4.2 sont calculés, mais nous ne considérons que les instances qui peuvent être résolues à l'optimalité par CPLEX pour la PLNE sans et avec le prétraitement.

Les résultats présentés dans les Tableaux 4.2 et 4.3 montrent que généralement la PLNE avec le prétraitement peut résoudre des instances plus grandes que la PLNE originale.

TABLE 4.2 – Comparaison entre le modèle d'origine de la PLNE et la PLNE avec le prétraitement sur le problème mono-critère (1)

Type	p	pc	nd_{min}	nd_{avg}	nd_{max}	t_{min}	t_{avg}	t_{max}
carré	2	100.0	227	64618.4	347444	0.3	45.7	330.9
		96.7	0	16390.9	152887	0.0	13.3	130.5
	3	66.7	796	85628.4	658183	1.9	144.6	1634.5
		70.0	193	96439.0	1058146	4.4	160.6	1666.0
	4	56.7	14137	120403.8	295361	72.7	238.1	496.3
		90.0	3	58111.4	378665	9.9	295.8	1355.1
rectangle	2	96.7	0	231867.4	1903935	0.0	165.6	1519.3
		100.0	0	39550.2	1113782	0.0	60.8	1676.7
	3	73.3	613	135976.3	718409	0.5	243.2	1637.0
		93.3	0	14778.5	298746	0.6	43.6	612.3
	4	60.0	1078	144623.3	585489	3.5	356.2	1708.7
		96.7	0	21483.8	301202	1.5	142.3	1197.0
peu	2	83.3	66	50999.2	236296	0.3	38.9	204.9
perturbé		100.0	0	5494.0	84459	0.0	6.4	48.0
	3	63.3	730	180910.7	781927	0.6	355.0	1677.3
		93.3	0	44076.5	739065	0.7	98.4	1506.9
	4	40.0	1095	88410.8	392136	3.8	260.2	1329.2
		93.3	0	49009.0	697339	2.3	182.5	1810.8
perturbé	2	100.0	487	11211.3	108141	0.3	13.1	68.7
		100.0	0	0.0	0	0.0	0.2	0.8
	3	93.3	669	35621.4	226787	0.7	115.6	557.7
		100.0	0	2254.3	34946	0.5	13.1	93.7
	4	66.7	379	37444.3	179166	0.6	181.4	670.8
		96.7	0	9219.9	90276	1.5	79.7	431.7

TABLE 4.3 – Comparaison entre le modèle d'origine de la PLNE et la PLNE avec le prétraitement sur le problème mono-critère (2)

type	p	nd_{min}	nd_{avg}	nd_{max}	t_{min}	t_{avg}	t_{max}
carré	2	227	56186.7	347444	0.3	36.5	330.9
		0	16390.9	152887	0.0	13.3	130.5
	3	796	56331.0	222279	1.9	64.4	289.3
		193	34374.7	581020	4.4	59.9	947.0
	4	14137	120403.8	295361	72.7	238.1	496.3
		3	758.5	2398	9.9	17.1	24.5
rectangle	2	0	231867.4	1903935	0.0	165.6	1521.9
		0	2507.8	56640	0.0	5.0	45.9
	3	613	135976.3	718409	0.5	243.2	1637.0
		0	267.8	1357	0.6	8.7	26.1
	4	1078	144623.3	585489	3.5	356.2	1708.7
		0	482.7	2143	1.5	17.1	35.9
peu perturbé	2	66	50999.2	236296	0.3	38.9	204.9
		0	5394.7	84459	0.0	5.3	48.0
	3	730	180910.7	781927	0.6	355.0	1677.3
		0	4461.7	58278	0.7	14.0	110.6
	4	1095	88410.8	392136	3.8	260.2	1329.2
		0	489.3	1485	2.3	12.9	45.0
perturbé	2	487	11211.3	108141	0.3	13.1	68.7
		0	0.0	0	0.0	0.2	0.8
	3	669	35621.4	226787	0.7	115.6	557.7
		0	8.4	134	0.5	8.5	48.0
	4	379	37292.5	179166	0.6	183.1	670.8
		0	474.9	5647	1.5	25.8	75.4

D'abord, à l'exception des cartes carrées avec 2 secteurs, la PLNE avec le prétraitement aide toujours à résoudre plus d'instances dans les limites imposées : le scénario le plus favorable est obtenue pour les cartes peu perturbées avec 4 secteurs où l'utilisation du prétraitement conduit à augmenter le pourcentage d'instances résolues de 40% à $93,3\%$ (voir le Tableau 4.2). En ce qui concerne le nombre de nœuds de l'algorithme de branch-and-cut appliquée par CPLEX pour résoudre les deux PLNE, le Tableau 4.3 montre que pour les instances les plus difficiles, c'est à dire les cartes carrées, le pourcentage de réduction du nombre de nœuds en moyenne est environ $70,8\%$, $39,0\%$, $99,4\%$ pour 2, 3, 4 secteurs respectivement. Par conséquence, le prétraitement a un impact fort sur la résolution exacte de la PLNE, ce qui est confirmé par le temps CPU requis par CPLEX. Le Tableau 4.3 montre encore que, pour les cartes carrées, le pourcentage de réduction du temps CPU en moyenne est d'environ $63,6\%$, $7,0\%$, $92,8\%$ pour 2, 3, 4 secteurs, respectivement. Et pour les cartes perturbées, le pourcentage de réduction du temps CPU est d'environ $98,5\%$, $92,6\%$, $85,9\%$ pour 2, 3, 4 secteurs respectivement. La comparaison des Tableaux 4.2 et 4.3 montre que la plupart du temps le prétraitement permet de résoudre des instances que CPLEX n'arrive pas à résoudre seul. Les exceptions sont les cartes carrés avec 2 secteurs. Il s'avère que, même en considérant toutes les instances, le temps CPU en moyenne et le nombre de nœuds nécessaires pour résoudre la PLNE avec le prétraitement sont meilleurs. Ceci renforce la conclusion que le prétraitement présenté aide significativement à résoudre le problème de sectorisation.

Nous considérons maintenant le problème bicritère et cherchons à évaluer l'influence des ϵ-contraintes et δ-contraintes sur la résolution du problème. Nous avons réalisé des tests sur les mêmes instances que pour le problème mono-critère. Nous fixons les valeurs de ϵ et δ comme suit : $\epsilon = 80\% \times \sum_{v_i \in G} w_i^1/p$, $\delta = 200\% \times \sum_{v_i \in G} w_i^1/p$. Les Tableaux 4.4 et 4.5 sont arrangés de la même manière que Tes tableaux 4.2 et 4.3. Ils montrent l'impact du prétraitement avec les inégalités valides sur la résolution de la PLNE. Les résultats présentés dans ces deux tableaux sont calculés sur toutes les instances, y compris celles pour lesquelles une solution optimale ne peux pas être trouvée. La raison pour laquelle nous ne trouvons pas une solution optimale peut être due à la limite sur le temps de calcul ou sur la taille de la mémoire, ou simplement, il peut ne pas exister de solution faisable à cause des ϵ-contraintes ou des δ-contraintes.

Les résultats présentés dans les Tableaux 4.4 et 4.5 montrent que la différence de performance entre la PLNE avec le prétraitement et la PLNE originale sur le problème bicritère est plus grande que celle sur le problème mono-critère. D'abord, dans tous les cas, la PLNE avec le prétraitement peut résoudre plus d'instances dans les limites imposées : les scénarios les plus favorables sont obtenus pour les cartes peu perturbées et perturbées avec 4 secteurs où l'utilisation du prétraitement conduit à augmenter le pourcentage d'instances résolues de $40,0\%$ à $96,7\%$, et de $26,7\%$ à $83,3\%$. En ce qui concerne le nombre de nœuds, le Tableau 4.5 montre que pour les instances avec des cartes carrées, le pourcentage de réduction du nombre de nœuds en moyenne est d'environ $94,1\%$, $97,7\%$, $98,3\%$ pour 2, 3, 4 secteurs respectivement ; pour les cartes rectangulaires, elle est d'environ $99,5\%$, $99,8\%$, $67,7\%$ respectivement ; pour les cartes peu perturbées, elle est d'environ $97,5\%$, $99,7\%$, $95,4\%$ respectivement ; et pour les cartes perturbées, elle est d'environ 100%, $99,9\%$, $92,7\%$ respectivement. Le Tableau 4.5 montre encore que, en ce qui concerne le temps de calcul, pour les cartes carrées le pourcentage de réduction du temps en moyenne est d'environ

TABLE 4.4 – Comparaison entre le modèle d'origine de la PLNE et la PLNE avec le prétraitement sur le problème bicritère (1)

Type	p	pc	nd_{min}	nd_{avg}	nd_{max}	t_{min}	t_{avg}	t_{max}
carré	2	100.0	0	82783.1	682767	0.2	60.8	630.1
		100.0	0	4851.4	27374	1.2	8.8	26.0
	3	60.0	11673	66253.1	276813	13.8	149.4	927.3
		73.3	0	2900.6	16141	8.6	48.4	205.6
	4	60.0	34411	108942.8	204323	173.3	729.7	1776.4
		76.7	0	15075.3	88135	29.5	474.5	2196.8
rectangle	2	100.0	225	126876.1	816270	0.2	115.3	978.8
		100.0	0	634.1	8730	0.3	7.2	51.7
	3	60.0	1074	73932.7	323980	3.3	164.5	814.6
		100.0	0	9260.0	93251	2.4	130.0	1632.5
	4	50.0	3828	50069.9	174087	9.2	250.6	1438.4
		93.3	0	14518.7	224834	7.4	296.9	2504.4
peu	2	83.3	0	78345.7	418370	0.1	56.3	321.4
perturbé		100.0	0	17593.7	476107	0.0	28.0	628.9
	3	60.0	778	107455.3	380395	1.8	295.4	1102.5
		100.0	0	7853.0	112025	1.8	87.1	540.2
	4	40.0	901	58522.1	220999	4.5	353.3	1770.2
		96.7	0	7236.4	64568	8.8	312.2	1377.7
perturbé	2	100.0	882	40299.3	551704	2.2	45.4	588.8
		100.0	0	0.0	0	0.1	0.8	4.9
	3	90.0	2664	73966.7	325423	16.4	312.5	1784.2
		100.0	0	1421.0	39918	7.8	47.0	199.7
	4	26.7	29003	63672.8	138103	150.6	429.9	803.6
		83.3	0	1657.8	18252	45.4	252.8	1003.7

$85,5\%$, $83,5\%$, $85,7\%$ pour 2, 3, 4 secteurs respectivement ; pour les cartes rectangles, elle est d'environ $93,8\%$, $84,1\%$, $23,4\%$ respectivement ; pour les cartes peu perturbées, elle est d'environ $89,5\%$, $92,2\%$, $56,6\%$ respectivement ; et pour les cartes perturbées, elle est d'environ $98,2\%$, $87,0\%$, $29,5\%$ respectivement.

4.2 Une procédure par séparation et évaluation

Dans cette section nous présentons les différents éléments d'une procédure par séparation et évaluation (Branch-and-Bound) permettant de résoudre optimalement le problème de sectorisation bicritère sans pôle prédéfini. L'objectif que nous poursuivons est d'essayer d'établir une méthode exacte dédiée qui soit plus efficace que la méthode arborescente générique utilisée par CPLEX pour résoudre le modèle de PLNE.

Notons que les méthodes exactes de type Branch-and-Bound sont très largement utilisées pour la résolution de problèmes d'optimisation combinatoire. Nous rappelons ici sché-

TABLE 4.5 – Comparaison entre le modèle d'origine de la PLNE et la PLNE avec le prétraitement sur le problème bicritère (2)

type	p	nd_{min}	nd_{avg}	nd_{max}	t_{min}	t_{avg}	t_{max}
carré	2	0	82783.1	682767	0.2	60.8	630.1
		0	4851.4	27374	1.2	8.8	26.0
	3	11673	66253.1	276813	13.8	149.4	927.3
		0	1495.4	16141	8.6	24.6	205.6
	4	34411	108942.8	204323	173.3	729.7	1776.4
		0	2003.5	33783	29.5	104.4	1044.7
rectangle	2	225	126876.1	816270	0.2	115.3	978.8
		0	634.1	8730	0.3	7.2	51.7
	3	1074	73932.7	323980	3.3	164.5	814.6
		0	178.3	1671	2.4	26.1	271.8
	4	3828	50078.6	174087	9.2	245.3	1438.4
		0	16153.3	224834	7.4	188.0	2195.6
peu perturbé	2	0	78345.7	418370	0.1	56.3	321.4
		0	1957.9	23333	0.0	5.9	23.7
	3	778	107455.3	380395	1.8	295.4	1102.5
		0	273.4	3796	1.8	23.0	126.9
	4	901	58522.1	220999	4.5	353.3	1770.2
		0	2697.0	29184	8.8	153.5	1377.7
perturbé	2	882	40299.3	551704	2.2	45.4	588.8
		0	0.0	0	0.1	0.8	4.9
	3	2664	73966.7	325423	16.4	312.5	1784.2
		0	80.3	1972	7.8	40.7	197.9
	4	29003	63672.8	138103	150.6	429.9	803.6
		0	4671.0	18252	105.9	303.0	1003.7

matiquement les principes généraux d'une procédure par séparation et évaluation avant de détailler les éléments de notre méthode. Nous renvoyons à [Tercinet, 2004] pour une étude précise des différents mécanismes utilisés dans la littérature.

Une procédure par séparation et évaluation est une méthode classique permettant d'énumérer implicitement toutes les solutions d'un espace de recherche correspondant à un problème combinatoire dans le but de trouver une solution optimale. Dans une procédure par séparation et évaluation, l'espace de recherche est représenté par un arbre où chaque nœud représente une partie de cet espace. L'ensemble des nœuds fils obtenu à partir d'un nœud père, par séparation, représente le même sous-espace de recherche que le nœud père. Par exemple, le nœud racine représente l'espace de recherche dans sa totalité. Par séparation du nœud racine on obtient un ensemble de nœuds fils tel que l'union des sous-espaces de recherche associés à chacun de ces nœud fils représente la totalité de l'espace de recherche. L'opération de séparation consiste donc à partitionner l'espace de recherche associé à un nœud, chaque sous partie étant représentée par un nœud fils. Passer d'un nœud père à un nœud fils correspond à prendre une des décisions qui doivent être prises pour la résolution du problème et donc réduire l'espace de recherche. Typiquement une décision consiste à réduire le domaine en fixant une ou plusieurs variables de décision du problème initial. Les nœuds feuilles de l'arbre de recherche correspondent à des solutions du problème où toutes les décisions ont été prises, c'est à dire que toutes les variables sont fixées.

La méthode stocke dans une structure, par exemple une pile, l'ensemble des nœuds qui correspondent à l'espace de recherche qui n'a pas encore été exploré. Sans perte de généralité nous considérons ici une méthode par séparation et évaluation pour la résolution optimale d'un problème de minimisation. Tant qu'il reste des nœuds dans la pile, le principe de fonctionnement de la méthode est le suivant :

- un nœud N est extrait de la pile,
- si N est un nœud feuille et que la solution qu'il représente a un coût strictement meilleur que la borne supérieure UB alors UB est mise à jour et la meilleure solution est stockée,
- si N n'est pas une feuille de l'arbre de recherche, une borne inférieure $LB(N)$ associée au nœud N est calculée, où $LB(N)$ est évaluation par défaut de la valeur de la meilleure solution contenue dans le sous-espace de recherche associé au nœud N,
- $LB(N)$ est comparée à la meilleure solution trouvée, UB. Si $LB(N) \geq UB$ alors le nœud est coupé. En effet dans ce cas, la meilleure solution accessible à partir de N a un coût au moins égal à celui de la meilleure solution connue, c'est à dire qu'il est impossible d'améliorer la meilleure solution connue à partir de N. Dans le cas contraire, on construit les nœuds fils du nœud N et ces nœuds sont ajoutés à la pile pour être explorés ultérieurement.

Une fois que la pile est vide, la meilleure solution trouvée au cours de la recherche est une solution de coût minimal. Dans les sections suivantes, nous décrivons les éléments de la procédure par séparation et évaluation spécifiques à notre problème : le schéma de séparation, la borne supérieure, la borne inférieure, la stratégie d'exploration de l'arbre de recherche et les conditions de dominance utilisées.

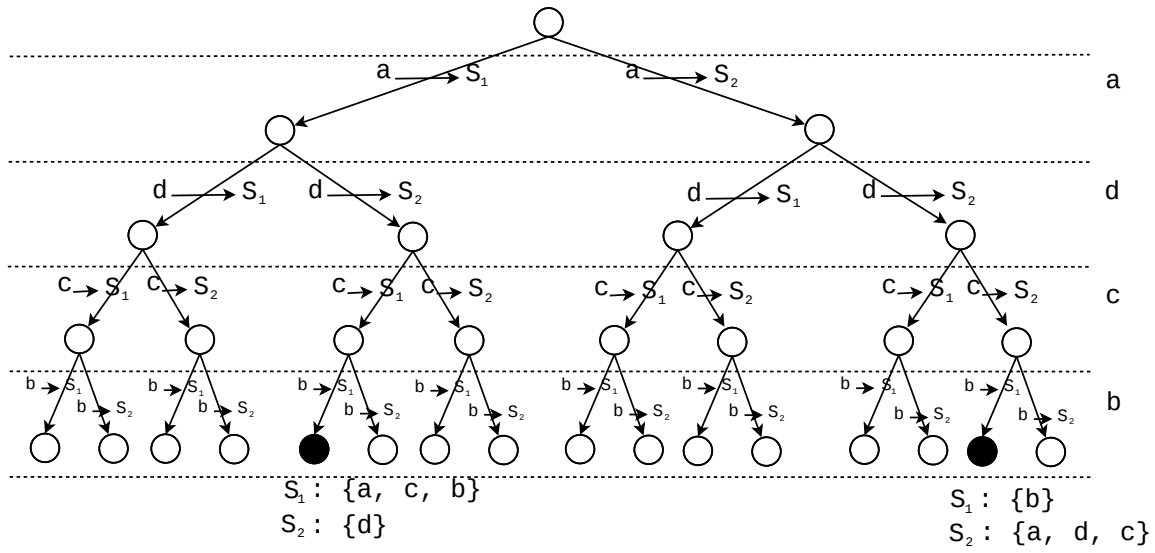

FIGURE 4.6 – Exemple d'arbre de recherche

4.2.1 Schéma de branchement

Le schéma de branchement définit la manière dont l'espace de recherche est partitionné lorsque l'on passe d'un nœud père à ses nœuds fils, c'est-à-dire quelles décisions sont prises entre le nœud père et les nœuds fils. Le plus souvent le schéma de branchement dépend de la représentation d'une solution.

Pour le problème qui nous intéresse ici, nous utilisons une manière de représenter une solution qui consiste à décider de l'affectation des éléments basiques aux secteurs. Nous classons l'ensemble des éléments V par leur poids décroissant sur le premier critère w_i^1 et nous obtenons une liste tiée des éléments $V = \{v_1, ..., v_n\}$. Le nombre de niveaux dans l'arbre de recherche est égale au nombre d'éléments à sectoriser, et le nombre de branchements entre le nœud père et les nœuds fils est égale au nombre de secteurs. Le nombre de nœuds de l'arborescence est donc en $O(|J|^{|I|})$. Pour un nœud intermédiaire (qui n'est pas une feuille) de niveau i, un de ses nœuds fils est créé en affectant l'élément v_i au secteur S_j. La Figure 4.6 présente un arbre de recherche utilisant ce schéma de branchement pour un problème a 4 éléments et 2 secteurs.

4.2.2 Borne supérieure

Dans une procédure par séparation et évaluation, la borne supérieure UB est utilisée pour fournir une solution faisable avant le début de l'exécution de la procédure. Les méthodes permettant d'obtenir des bornes supérieures sont des heuristiques qui en général permettent d'obtenir rapidement une solution faisable. Si la solution obtenue est de bonne qualité, la méthode peut éventuellement couper plus de nœuds dès le début de l'exécution.

Une heuristique peut être utilisée sur un nœud racine ou sur un nœud qui est en cours d'être traité. Dans le deuxième cas, l'heuristique construit rapidement une solution en

fixant les affectations non encore fixées dans le nœud traité, c'est à dire que pour un nœud de niveau i, l'heuristique trouve les affectations pour les éléments d'indice i à n selon une règle, si la solution obtenue est meilleure que la UB courante, la UB sera mise à jour.

L'heuristique que nous utilisons est HX_2 présentée dans la Section 3.2.1. Cette heuristique est uniquement utilisée au nœud racine et nous donne une UB initiale.

4.2.3 Borne inférieure

La borne inférieure LB sert à couper des nœuds dans l'arbre de recherche, en comparant une estimation par défaut de la meilleure solution accessible à partir de ce nœud à la meilleure solution connue. Le but est de détecter au plus tôt des zones de l'espace de recherche qui ne peuvent conduire à aucune solution meilleure que la borne supérieure courante (la meilleure solution trouvée).

Nous utilisons simplement la relaxation linéaire du modèle de PLNE donné dans la Section 4.1.1 pour calculer une LB initiale sur le nœud racine. La raison pour laquelle nous ne calculons pas la borne inférieure sur chaque nœud est quand le nombre de nœuds est très grand, il est très coûteux pour tous calculer.

4.2.4 Stratégie d'exploration

La stratégie d'exploration définit dans quel ordre les nœuds sont explorés. Trois stratégies sont classiquement utilisées :

- *profondeur d'abord* : cette stratégie explore en priorité les nœuds de plus grands niveaux dans l'arbre de recherche. L'avantage de cette stratégie est d'explorer rapidement des nœuds feuilles correspondant à des solutions. De plus, cette stratégie utilise moins de mémoire que les autres car un seul nœud est développé par niveau de l'arbre de recherche. L'inconvénient de cette méthode est qu'elle peut explorer de manière intensive de petites zones de l'espace de recherche qui ne contiennent pas de solution optimale.
- *largeur d'abord* : cette stratégie explore en priorité les nœuds de plus petit niveau dans l'arbre de recherche. L'avantage de cette stratégie est de pouvoir couper tôt des nœuds qui représentent des sous-espace important, mais cela n'est possible que si l'on dispose d'une borne supérieure de bonne qualité. L'inconvénient majeur de cette stratégie est d'utiliser un espace mémoire important puisque tous les nœuds d'un niveau peuvent être développés simultanément.
- *meilleur d'abord* : cette stratégie explore en priorité les nœuds de plus petites bornes inférieures et parmi les nœuds de plus petites bornes inférieures, ceux de plus grand niveau dans l'arbre de recherche. L'inconvénient de cette stratégie est qu'il faut calculer toutes les bornes inférieures des nœuds présents dans la pile qui est très coûteux en temps de calcul. En plus cette stratégie peut aussi utiliser un espace mémoire important.

Nous utilisons la stratégie en profondeur d'abord. En effet, l'utilisation d'une stratégie en largeur d'abord ou en meilleur d'abord explose la taille de la pile et n'est donc pas utilisable. Néanmoins, [T'kindt *et al.*, 2004] ont montré que une utilisation judicieuse de stratégies de comparaison de nœuds, des conclusion inverses pouvaient être obtenues pour

certains problèmes d'ordonnancement.

4.2.5 Conditions de dominance

Une condition de dominance est une condition permettant de déduire qu'un nœud N_1 domine un autre N_2 : cela signifie que la meilleure solution de l'espace de recherche représenté par N_1 sera nécessairement meilleure ou égale à la meilleure solution de l'espace de recherche représenté par N_2. Dans une procédure par séparation et évaluation, cette condition est utilisée afin de couper des nœuds dominés par d'autres et ainsi réduire la taille de l'espace de recherche à explorer. Dans cette section nous présentons quelques conditions de dominance pour notre problème. Ces conditions de dominance sont basées sur les inégalités valides présentées dans la Section 4.1.2.

Dans l'arbre de recherche, un nœud N_i de niveau i correspond à une solution partielle ou entière. Nous développons quelques conditions de dominance pour l'éliminer.

Condition de dominance 1 :
Pour un nœud N_i, s'il existe $j \in J$ tel que

$$d_j^1 \geq e_j^1 + UB, \tag{4.24}$$

ce nœud peut être éliminé. d_j^1 est la valeur, sur le premier critère, du secteur S_j au nœud N_i, et e_j^1 est la valeur de référence du secteur S_j.

Cette condition de dominance correspond au Lemme 4.1.1. Nous pouvons aussi utiliser les δ-contraintes (4.8) qui imposent que la valeur de chaque secteur du nœud N_i sur le deuxième critère, notée par d_j^2, ne peut pas dépasser la valeur δ. S'il existe $j \in J$ tel que $d_j^2 > \delta$ pour un nœud N_i, nous pouvons éliminer ce nœud.

Condition de dominance 2 :
Ensuite, nous prenons un petit exemple pour introduire une autre condition de dominance. Si nous avons 4 éléments $V = \{v_1, v_2, v_3, v_4\}$ à mettre dans 2 secteurs, et deux solutions M et M'. Soit S_M^j et $S_{M'}^j$ l'ensemble des éléments dans le secteur S_j des solutions M et M', respectivement. Supposons que ces deux solutions soient : $S_M^1 = \{v_1, v_2\}$, $S_M^2 = \{v_3, v_4\}$; $S_{M'}^1 = \{v_3, v_4\}$, $S_{M'}^2 = \{v_1, v_2\}$. Ces deux solutions sont identiques sauf que le nom de chaque secteur est différent. Pour éliminer cette redondance de solutions pendant le parcours de l'arbre de recherche, nous faisons un tri sur les valeurs de référence e_j dans un ordre non-décroissant.

Pour un nœud feuille qui est descendant du nœud N_i, nous notons f_j^1 sa valeur sur le premier critère du secteur S_j. Pour le nœud N_i, d_j^1 est sa valeur sur le premier critère du secteur S_j et nous notons r_j^1 la valeur sur le premier critère qui reste à remplir dans le secteur S_j pour qu'il devient une solution (un nœud feuille). Donc on a : $f_j^1 = d_j^1 + r_j^1$. Pour définir l'intervalle de r_j^1, nous notons b_j^L et b_j^U ses bornes inférieure et supérieure.

Chaque nœud alors doit satisfaire les conditions suivantes :

$$b_j^L \leq r_j^1 \leq b_j^U ;\ b_j^L = e_j^1 - UB - d_j^1 ;\ b_j^U = e_j^1 + UB - d_j^1 \quad \forall j \tag{4.25}$$

$$r_j^1 + d_j^1 \leq r_{j+1}^1 + d_{j+1}^1 \quad \forall j = 1, ..., p-1 \tag{4.26}$$

Algorithme de vérification de la condition de dominance 2
Pour j de 1 à $p-1$ **faire**
$b_j^L = e_j^1 - UB - d_j^1$;
$b_j^U = e_j^1 + UB - d_j^1$;
Si $(b_j^U < 0)$, **Alors**
couper le nœud testé ;
Finsi
Si $(b_j^L < 0)$, **Alors**
$b_j^L = 0$;
Finsi
Fin Pour
Pour j de 1 à p **faire**
Si $(b_{j+1}^U + d_{j+1}^1 - d_j^1 < b_j^L)$, **Alors**
couper le nœud testé ;
Finsi
Si $(b_{j+1}^U + d_{j+1}^1 - d_j^1 < b_j^U)$, **Alors**
$b_j^U = b_{j+1}^U + d_{j+1}^1 - d_j^1$
Si $(b_j^L + d_j^1 - d_{j+1}^1 > b_{j+1}^U)$, **Alors**
couper le nœud testé ;
Finsi
Si $(b_j^L + d_j^1 - d_{j+1}^1 > b_{j+1}^L)$, **Alors**
$b_{j+1}^L = b_j^L + d_j^1 - d_{j+1}^1$
Fin Pour
Si $(\sum_{j=1}^p b_j^L > \sum_{j=1}^p e_j^1 - \sum_{j=1}^p d_j^1$ ou $\sum_{j=1}^p b_j^U < \sum_{j=1}^p e_j^1 - \sum_{j=1}^p d_j^1$ $)$, **Alors**
couper le nœud testé.

FIGURE 4.7 – Algorithme de vérification de la condition de dominance 2

$$\sum_{j=1}^{p} r_j^1 = \sum_{j=1}^{p} e_j^1 - \sum_{j=1}^{p} d_j^1 \tag{4.27}$$

La condition 4.25 vient du Lemme 4.1.1 tandis que la condition 4.26 correspond à la Proposition 2. La condition 4.27 vient du fait que la somme des valeurs des éléments déjà affectés plus la somme des valeurs restantes doit être égale à la valeur totale. A partir de (4.25) et (4.26), nous pouvons déduire ces deux inégalités :

$$b_{j+1}^U \geq r_{j+1}^1 \geq r_j^1 + d_j^1 - d_{j+1}^1 \quad \forall j = 1, ..., p-1 \tag{4.28}$$

$$b_j^L \leq r_j^1 \leq r_{j+1}^1 + d_{j+1}^1 - d_j^1 \quad \forall j = 1, ..., p-1 \tag{4.29}$$

L'algorithme qui permet de vérifier ces inégalités est donné dans la Figure 4.7. Son application permet également de vérifier la condition de dominance 1. Sa complexité est en $O(p)$.

Condition de dominance 3 :
Nous pouvons utiliser le Lemme 4.1.2 pour introduire une autre condition de dominance.

Pour un nœud N_i de niveau i pour lequel l'élément courant v_i vient d'être affecté au secteur S_j, si la valeur sur le premier critère de cet élément w_i^1 n'est pas assez grande pour que le secteur S_j ($w_i^1 < e_j^1 - UB$) et si tous ses voisins sont déjà affectés dans un autre secteur que S_j, alors le nœud N_i peut être éliminé.

Condition de dominance 4 :
De la Proposition 3, nous pouvons déduire également une condition de dominance. Pour un nœud N_i de niveau i, nous nous intéressons à l'ensemble des éléments R qui sont déjà affectés dans un secteur S_j, $\forall j \in J$, et nous notons t_R le coût sur le premier critère de l'arbre de Steiner pour connecter tous les éléments R dans S_j.

Pour un nœud N_i, s'il existe $j \in J$ tel que $t_R > e_j^1 + UB$, alors ce nœud peut être éliminé.

Comme le problème pour calculer le coût, t_R, d'un arbre de Steiner est NP-difficile, dans notre PSE, nous calculons les coûts pour tous les ensembles de terminaux R telle que $|R| \leq 3$ avec la méthode de programmation dynamique (Section 4.1.2). Pour les ensembles $R \subseteq S_j$ tels que $|R| > 3$, nous posons $t_R = \sum_{i \in R} w_i^1$. Évidemment, dans ce dernier cas, cette façon de calcul en t_R est très simple et il reste du travail pour l'améliorer : par exemple, en utilisant des bornes inférieures pour le problème de l'arbre de Steiner.

Condition de dominance 5 :
De la Proposition 4, nous pouvons déduire également une condition de dominance. Supposons que nous ayons calculé, à l'initialisation de la PSE, tous les séparateurs Sep comme indiqué dans la Section 4.1.2. Pour chaque séparateur Sep_l, nous notons par C_k^l ($k \in \{1, ..., K_l\}$) l'ensemble des composants connexes que sépare Sep_l.

Pour un nœud N_i de niveau i pour lequel l'élément v_i vient d'être affecté au secteur S_j, si $\exists Sep_l$ tel que $v_i \in C_k^l$ et si la valeur sur le premier critère de ce composant $f_{C_k}^1 < e_j^1 - UB$, et si tous les éléments du séparateur Sep_l sont déjà affectés dans un autre secteur que S_j, alors le nœud N_i peut être éliminé.

Condition de dominance 6 :
Nous pouvons aussi vérifier la connexité de chaque secteur, pour un nœud N_i, correspondant à une solution partielle ou entière. Si nous pouvons détecter un secteur qui n'est pas connexe, alors le nœud N_i est coupé.

Pour un nœud N_i, entre chaque paire d'éléments affectés dans un même secteur, s'il n'existe pas de chemin reliant ces deux en n'utilisant que des éléments déjà affectés dans ce secteur où des éléments non encore affectés, alors ce nœud peut être éliminé.

4.2.6 Les résultats expérimentaux

Dans cette section, nous présentons les tests expérimentaux pour évaluer les performances de la procédure par séparation et évaluation présentée en comparant avec la résolution du modèle de PLNE avec un prétraitement initiale présenté dans la Section 4.1.3. Les tests sont lancés sur les mêmes instances avec celles qui évaluent l'efficacité de l'algorithme de prétraitement sur le problème de sectorisation bicritère. Nous fixons les valeurs de ϵ et δ : $\epsilon = 80\% \times \sum_{v_i \in G} w_i^1/p$, et $\delta = 200\% \times \sum_{v_i \in G} w_i^1/p$. Pour la résolution de la PLNE avec prétraitement, dans le solveur CPLEX, nous fixons toujours des limites sur la mémoire utilisée par l'arbre de recherche à 650 Mo et sur le temps de calcul à 1800 secondes

TABLE 4.6 – Comparaison entre la PLNE avec prétraitement et la PSE sur le problème bicritère (1)

Type	p	pc	nd_{min}	nd_{avg}	nd_{max}	t_{min}	t_{avg}	t_{max}
carré	2	100.0	0	4851.4	27374	1.2	8.8	26.0
		100.0	3784	208342.7	654758	0.0	0.3	1.1
	3	73.3	0	2900.6	16141	8.6	48.4	205.6
		56.7	1055130	1385881.9	1893147	0.8	1.1	1.5
	4	76.7	0	15075.3	88135	29.5	474.5	2196.8
		56.7	193877176	240476096.5	306267516	181.7	230.3	296.7
rectangle	2	100.0	0	634.1	8730	0.3	7.2	51.7
		100.0	304	60167.7	601662	0.0	0.1	1.3
	3	100.0	0	9260.0	93251	2.4	130.0	1632.5
		90.0	42399	75136724.3	671366715	0.0	71.2	644.3
	4	93.3	0	14518.7	224834	7.4	296.9	2504.4
		43.3	2112164	91359809.2	266559820	2.0	86.0	261.2
peu	2	100.0	0	17593.7	476107	0.0	28.0	628.9
perturbé		100.0	108	48856.7	364492	0.0	0.1	0.8
	3	100.0	0	7853.0	112025	1.8	87.1	540.2
		86.7	38259	135082969.7	1236603126	0.0	129.1	1214.9
	4	96.7	0	7236.4	64568	8.8	312.2	1377.7
		36.7	2610816	207958196.0	773336840	3.1	199.2	727.5
perturbé	2	100.0	0	0.0	0	0.1	0.8	4.9
		100.0	436	1822.6	9672	0.0	0.0	0.0
	3	100.0	0	1421.0	39918	7.8	47.0	199.7
		96.7	9968616	185050607.5	1468473507	8.6	180.5	1477.6
	4	83.3	0	1657.8	18252	45.4	252.8	1003.7
		0.0						

TABLE 4.7 – Comparaison entre la PLNE avec prétraitement et la PSE sur le problème bicritère (2)

type	p	nd_{min}	nd_{avg}	nd_{max}	t_{min}	t_{avg}	t_{max}
carré	2	0	4851.4	27374	1.2	8.8	26.0
		3784	208342.7	654758	0.0	0.3	1.1
	3	0	633.9	1781	8.6	14.0	23.6
		1055130	1385881.9	1893147	0.8	1.1	1.5
	4	0	134.1	256	29.5	49.1	78.2
		193877176	240476096.5	306267516	181.7	230.3	296.7
rectangle	2	0	634.1	8730	0.3	7.2	51.7
		304	60167.7	601662	0.0	0.1	1.3
	3	0	4863.3	93251	2.4	55.8	467.1
		42399	75136724.3	671366715	0.0	71.2	644.3
	4	0	95.4	706	7.4	28.8	74.9
		2112164	91359809.2	266559820	2.0	86.0	261.2
peu perturbé	2	0	17593.7	476107	0.0	28.0	628.9
		108	48856.7	364492	0.0	0.1	0.8
	3	0	5492.8	112025	1.9	56.6	306.8
		38259	135082969.7	1236603126	0.0	129.1	1214.9
	4	0	334.2	1963	8.8	31.5	77.9
		2610816	207958196.0	773336840	3.1	199.2	727.5
perturbé	2	0	0.0	0	0.1	4.9	0.8
		436	1822.6	9672	0.0	0.0	0.0
	3	0	93.5	1972	7.8	45.4	197.9
		9968616	185050607.5	1468473507	8.6	180.5	1477.6
	4	0	1657.8	18252	45.4	1003.7	252.8

et pour la procédure par séparation et évaluation, nous fixons aussi une limite sur le temps de calcul à 1800 secondes.

Les Tableaux 4.6 et 4.7 sont arrangés de la façon suivante : pour chaque type de carte et chaque valeur de p, la première ligne présente les résultats correspondant à la résolution de la PLNE avec un prétraitement initiale tandis que la deuxième ligne correspond à la procédure par séparation et évaluation. Le Tableau 4.6 présente le pourcentage d'instances résolues à l'optimalité (pc) dans les limites imposées, les nombres de nœuds minimal (nd_{min}), moyen (nd_{avg}) et maximal (nd_{max}) explorés par les méthodes arborescentes, et le temps CPU minimal (t_{min}), moyen (t_{avg}) et maximal (t_{max}) en secondes, nécessaire pour résoudre les problèmes optimalement. Dans le Tableau 4.7, les mêmes résultats que dans le Tableau 4.6 sont calculés, mais nous ne considérons que les instances qui peuvent être résolues à l'optimalité par CPLEX pour la procédure par séparation et évaluation et par la résolution du modèle de PLNE avec le prétraitement.

Les résultats présentés dans ces deux tableaux sont calculés sur toutes les instances, y compris celles pour lesquelles une solution optimale ne peux pas être trouvée. La raison pour laquelle nous ne trouvons pas une solution optimale peut être due à la limite sur le temps de calcul ou sur la taille de la mémoire, ou simplement, s'il n'existe pas de solution faisable à cause des ϵ-contraintes ou des δ-contraintes.

Les résultats présentés dans les Tableaux 4.6 et 4.7 montrent que généralement la PLNE avec prétraitement peut résoudre les instances plus grandes que la PLNE originale. Nous pouvons remarquer que le nombre de secteur p influence beaucoup la performance de la procédure par séparation et évaluation, quand p augmente, le problème devient beaucoup plus difficile à résoudre. Et contrairement au fait que la carte carrée est le plus difficile à résoudre pour la PLNE avec le prétraitement, nous ne pouvons pas avoir cette conclusion au niveau de la forme de la carte pour la procédure par séparation et évaluation. En ce qui concerne le nombre de nœuds, le tableau 4.7 montre que pour tous les cas, la procédure par séparation et évaluation nécessite d'explorer beaucoup plus de nœuds que la PLNE avec le prétraitement pour trouver une meilleure solution. En ce qui concerne le temps de calcul, quand le nombre de secteur p est petit, la procédure par séparation et évaluation mets toujours moins de temps, mais quand p augmente, c'est la PLNE avec le prétraitement qui est plus rapide pour trouver une meilleure solution.

4.3 Conclusion

Dans ce chapitre, nous avons présenté des méthodes de résolution exacte permettant de résoudre le problème de sectorisation sans pôle prédéfini.

Nous avons tout d'abord présenté une formulation mathématique de ce problème de sectorisation. Et puis nous ajoutons quelques inégalités valides pour restreindre l'espace des solutions et développons un algorithme de prétraitement pour réduire le nombre de variables. Cette méthode a fait l'objet d'une communication à la conférence nationale *la Société Française de Recherche Opérationnelle et d'Aide à la Décision* (ROADEF2012)[Tang *et al.*, 2012b] et d'une conférence internationale *Operation Research Peripatetic Postgraduate Programme* (ORP3)[Tang *et al.*, 2012a]. Une publication a été soumis et est dans l'état de révision majeure dans la revue internationale (*Annals of Operations Research* [Tang *et al.*, 2011b]).

Ensuite, nous présentons les éléments d'une procédure par séparation et évaluation. Le schéma de branchement choisi consiste à affecter les éléments aux secteurs à chaque nœud. La borne supérieure utilisée est l'heuristique HX_2 introduite au chapitre précédent. Nous présentons également 6 conditions de dominance permettant de réduire l'espace de recherche à explorer. Cette méthode a fait l'objet d'une conférence internationale *OR 2011* [Tang *et al.*, 2011a].

Enfin, nous présentons les résultats expérimentaux de ces méthodes. Ces résultats nous permettent de mettre en évidence les performances des différents méthodes de résolution. Concernant la procédure par séparation et évaluation, il reste encore du travail à faire, par exemple, le renforcement des conditions de dominance et la proposition d'une borne inférieure efficace et rapide. Cette méthode est à l'heure actuelle dans une version préliminaire.

Chapitre 5

Solution logicielle

5.1 Introduction

Dans les chapitres 3 et 4, nous avons présenté les méthodes de résolution que nous avons développé pour résoudre différents problèmes de sectorisation. Dans ce dernier chapitre, nous allons présenter la solution logicielle de sectorisation qui est un des modules de Cartes & Données 6 développé par la société Articque. Le module de sectorisation intègre l'heuristique H_2 que nous avons proposé dans la Section 3.1. Cette heuristique est développée pour résoudre le problème de sectorisation bicritère à partir de pôles donnés.

Nous allons, dans un premier temps, introduire la description et l'interface de ce module. Nous allons ensuite présenter une application de notre méthode de sectorisation pour le partage du territoire français en 6 secteurs. Nous montrons que les résultats obtenus sont de qualité en répondant aux contraintes de connexité et d'équilibre.

À la fin de ce chapitre, nous allons présenter brièvement les descriptions et les fonctionnements des autres modules du logiciel Cartes & Données 6 que j'ai développé pendant ma thèse CIFRE au sien de l'entreprise Articque.

5.2 Description du module de sectorisation

Le module de sectorisation que nous avons implémenté dans le logiciel Cartes & Données 6 résout le problème de sectorisation bicritère à partir de pôles donnés. Ce module de sectorisation se base, pour construire les secteurs, sur un maillage géographique (exemple : communes, cantons, départements, etc), et des données statistiques qui serviront de valeur à équilibrer ou de contrainte (exemple : chiffre d'Affaire, nombre de clients, population, etc). Le principe est de créer des groupes de territoires (secteurs) les plus équilibrés possibles par rapport à une donnée statistique fournie, et en tenant compte de contraintes imposées, comme les ϵ-contraintes et les δ-contraintes sur une autre donnée statistique.

La sectorisation s'effectue à partir de pôles qui sont fournis par l'utilisateur. En prenant ces pôles comme point de départ des secteurs, elle va étendre le territoire de chaque secteur en y associant les objets voisins du fond de carte, de manière à répartir le plus équitablement possible une donnée (le premier critère à optimiser) entre les différents secteurs et en

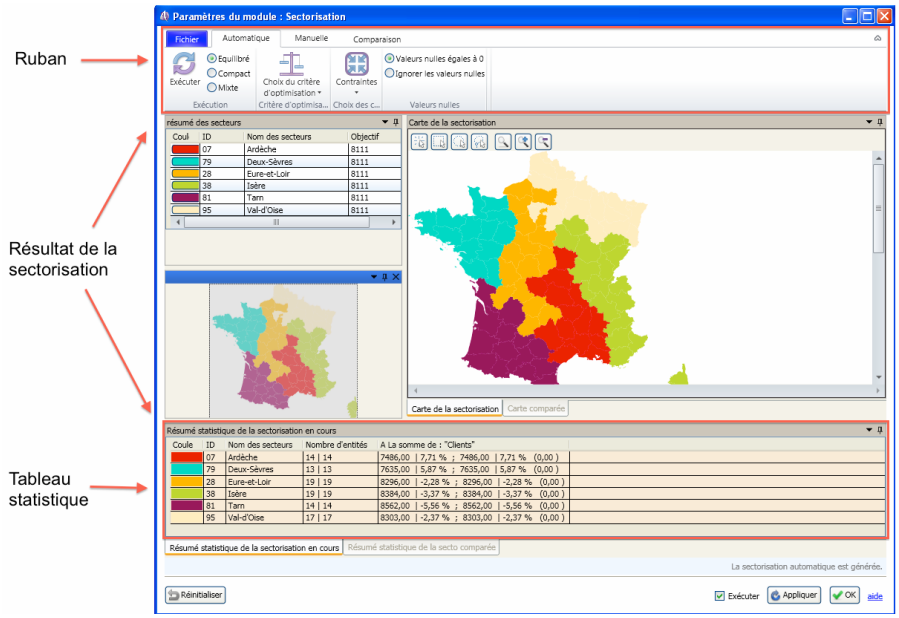

FIGURE 5.1 – Interface du module de sectorisation

respectant les contraintes imposées (les ϵ-contraintes et les δ-contraintes).

Pour effectuer cette sectorisation, le module nécessite en entrée :

- Des pôles issus du fond de carte servant de base à la création des secteurs (des objets du fond de carte). Ces pôles peuvent facilement être créés à l'aide d'un module "Sélection de pôles" relié au fond de carte.
- Une donnée continue à équilibrer : cette donnée représente le premier critère à optimiser. Elle devra être répartie et équilibrée entre les secteurs (ex : chaque secteur devra s'efforcer de contenir le même nombre d'habitants si la donnée utilisée est la population). Cette donnée continue doit être associée au fond de carte qui définit le maillage géographique pour la sectorisation.
- Une donnée continue de contrainte : cette seconde donnée représente le deuxième critère à optimiser (exprimé par les ϵ-contraintes, par exemple, l'écart maximum entre les secteurs sur la densité de population) et d'autre contraintes éventuellement (exprimé par les δ-contraintes, par exemple, les valeurs des secteurs sur la densité de population). Cette donnée continue doit être associée au fond de carte qui définit le maillage géographique pour la sectorisation.

Une fois que toutes les entrées sont prêtes pour le module de sectorisation, nous pouvons ouvrir l'interface dédiée à la création et aux modifications de la sectorisation. Nous pouvons trouver les différents composants de l'interface présentée dans la Figure 5.1 :

1. En haut de cette interface se trouve les menus Rubans. Ils contiennent les options et réglages de la sectorisation. C'est ici que l'utilisateur choisit le premier critère à optimiser. L'utilisateur peut éventuellement fixer les valeurs des contraintes sur d'autres données statistiques qui seront respectées dans le module lors de la sectorisation : l'écart maximum entre deux secteurs (le deuxième critère à minimiser, exprimé par

les ϵ-contraintes) et la valeur maximum d'un secteur (exprimé par les δ-contraintes).

2. La partie gauche au milieu de l'interface présente un tableau de résumé des secteurs et une carte de situation. Le tableau affiche les identifiants et les noms des secteurs choisis, ainsi que les valeurs de référence à atteindre. La carte de situation permet de visualiser la zone affichée et s'affiche également, en cas de zoom, dans la carte principale.
3. La partie droite au milieu de l'interface est la carte principale qui présente la sectorisation et les secteurs actuellement créés. On peut les modifier éventuellement, à la main, le résultat de la sectorisation.
4. En bas de cette interface, se trouve le tableau statistique des secteurs qui permet de visualiser la valeur de chaque secteur sur le premier critère et la déviation en pourcentage pour chaque secteur sur ce critère. Cette déviation, en pourcentage, mesure la différence entre la valeur réelle f_j^1 et la valeur de référence e_j^1 pour le secteur s_j qui est définie par $((f_j^1 - e_j^1)/e_j^1) \times 100$.

Cette interface est développé par un stagiaire Tingting Shan en utilisant le langage C# et la spécification graphique WPF (Windows Presentation Foundation), dans l'environnement Microsoft Visual Studio 2010.

5.3 Une application

Dans cette partie, nous prenons une application en utilisant le module de sectorisation à partir de centres. Dans cet exemple, nous avons la carte de la France et les objets géographiques (les éléments basiques) sont les départements. Pour chaque département, nous disposons des données statistiques comme le nombre de client et le nombre de prospects. Nous voulons sectoriser le territoire français en 6 secteurs tel que chaque secteur possède un nombre de clients égale, Dans cet exemple, cette valeur est égale à 8111. Nous avons également les contraintes suivantes :

- Nous sélectionnons les 6 pôles suivant Ardèche, Deux-Sèvres, Eure-et-loir, Isère, Tarn et Val-d'Oise.
- Chaque secteur doit être connexe.
- Concernant les ϵ-contraintes et les δ-contraintes, nous fixons leurs valeurs comme suit : $\epsilon = 100\% \times \sum_{v_i \in G} w_i^2/p$, $\delta = 150\% \times \sum_{v_i \in G} w_i^2/p$. Pour chaque élément v_i dans le graphe G, w_i^2 est le nombre de prospects de l'élément v_i, p est le nombre de secteurs qui égale à 6. Les ϵ-contraintes expriment que nous voulons aussi minimiser l'écart maximum entre deux secteurs sur le nombre de prospects, et nous bornons cet écart par la valeur de ϵ. Les δ-contraintes expriment que nous voulons également que le nombre de prospects de chaque secteur ne sont pas plus grand que la valeur de δ.

En tenant compte de ces contraintes et de l'objectif à atteindre, nous obtenons la solution issue du module de sectorisation. La solution sur la carte est affichée dans la partie droite au milieu de l'interface dans la Figure 5.1. Concernant les données statistiques, nous avons 7486 clients et 38155 prospects pour le secteur Ardèche, 7635 clients et 38259 prospects pour Deux-Sèvres, 8296 clients et 57074 prospects pour Eure-et-loir, 8384 clients

et 42943 prospects pour Isère, 8562 pour Tarn et 8303 clients et 50729 prospects pour Val-d'Oise. Donc dans cette solution, l'écart maximal est de $7,71\%$ à cause du secteur Ardèche et l'écart moyen est de $4,53\%$.

5.4 Développement des autres modules

Dans le logiciel Cartes & Données 6, nous avons différents modules qui réalisent chacun une fonctionnalité propre. Pendant ma thèse au sein de l'entreprise Articque, j'ai fait aussi le développement des modules suivants : "Union", "Longueur", "Surface", "Intersection", "Inclusion/Exclusion", "Distance minimum", "Fusion" et "Simulation d'implantation". Les descriptions et le fonctionnement de ces modules sont brièvement présentés ici.

Le module "Union" permet de réunir plusieurs fonds de carte pour n'en former qu'un seul qui correspond à leur assemblage.

Le module "Longueur" calcule la longueur de chaque ligne du fond de carte (par exemple : la longueur des routes), ainsi que le périmètre de chaque surface (par exemple : le périmètre des départements en France).

Le module "Surface" calcule la superficie de chaque entité géographique de type polygone du fond de carte.

Le module "Intersection" permet de sélectionner des entités géographiques d'un fond de carte qui sont "traversées" (intersection) par les entités géographiques d'un autre fond de carte. En d'autres termes, l'intersection de 2 fonds de cartes superposés géographiquement correspond à toutes les entités du premier fond de cartes qui "touchent" celle du second. Par exemple, ce module permet de sélectionner toutes les communes d'un fond de carte qui sont "traversées" par un réseau routier, de sélectionner les quartiers d'une ville dans lesquels se trouvent des points de clients géolocalisés, et de sélectionner tous les codes postaux d'une région qui sont "en dessous" d'une zone isochrone ou d'une zone géographique.

Le module "Inclusion/Exclusion" permet de sélectionner des entités géographiques d'un fond de carte qui sont "à l'intérieur/extérieur" (géographiquement parlant) des entités d'un autre fond de carte. Le mode "Inclusion" permet de récupérer les entités d'un fond de carte incluses dans les entités d'un autre fond de carte et le mode "Exclusion" permet de récupérer les entités d'un fond de carte qui sont en dehors des entités d'un autre fond de carte.

Le module "Distance minimum" calcule la distance la plus courte à vol d'oiseau qui sépare chaque entité géographique d'un fond de carte "A" de son entité la plus proche dans un fond de carte "B".

Le module "Fusion" permet la création de nouvelles zones géographiques en "assemblant" (opération de fusion) les entités d'un fond de carte existant.

Le module "Simulation d'implantation" a pour but d'estimer la viabilité de différents territoires d'implantation choisis, en calculant au choix : (1) la valeur totale d'une ou plusieurs données chiffrées dans un rayon de distance défini autour de chaque territoire d'implantation choisi. Par exemple, "Quelle est la population dans un rayon de 15 km autour de chacun de mes points de vente ?" ; (2) Le rayon de distance nécessaire à l'atteinte d'un objectif chiffré sur une donnée chiffré autour de chaque territoire d'implantation choisi.

Par exemple, "Quel territoire (rayon) autour de mon magasin faut-il définir pour atteindre 500 clients ?".

5.5 Conclusion

Dans ce chapitre, nous avons présenté la solution logicielle de sectorisation qui est un des modules de Cartes & Données 6. Et nous avons introduit la description et l'interface de ce module et ensuite nous avons présenté une application de notre méthode de sectorisation pour le partage du territoire français en 6 secteurs.

À part du module de sectorisation, dans le logiciel Cartes & Données 6, nous avons présenté brièvement les descriptions et les fonctionnements des modules "Union", "Longueur", "Surface", "Intersection", "Inclusion/Exclusion", "Distance minimum", "Fusion" et "Simulation d'implantation".

Nous avons aussi implémenté l'algorithme des boîtes dans le module de sectorisation, mais vu que le nombre de solutions non-dominées est très limité, nous n'avons pas mis cette méthode dans le logiciel Cartes & Données 6.

Si j'avais plus de temps à consacrer au module de sectorisation, j'aurais implémenté l'algorithme HX_2 pour résoudre le problème de sectorisation sans pôles prédéfini et résoudre d'autre types de problèmes de sectorisation, $P3$ par exemple.

-

Bibliographie

[Armstrong, 1974] Armstrong, R. (1974). Improved penalty calculations for a mixed integer branch-and-bound algorithm. *Mathematical Programming*, 6:212–223.

[Balas et Carrera, 1996] Balas, E. et Carrera, M. (1996). A dynamic subgradient-based branch-and-bound procedure for set covering. *Operations Research*, 44:875–890.

[Baldacci *et al.*, 2002] Baldacci, R., Hadjiconstantinou, E., Maniezzo, V. et Mingozzi, A. (2002). A new method for solving capacitated location problems based on a set partitioning approach. *Computers and Operations Research*, 29:365–386.

[Baptiste *et al.*, 2010] Baptiste, P., Della Croce, F., Grosso, A. et T'kindt, V. (2010). Sequencing a single machine with due dates and deadlines : an ILP-based approach to solve very large instances. *Journal of scheduling*, 13(1):39–47.

[Barnard et Simon, 1994] Barnard, S. et Simon, H. (1994). A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency : Practice and Experience*, 6:101–107.

[Beasley, 1987] Beasley, J. (1987). An algorithm for set covering problems. *European Journal of Operational Research*, 31:85–93.

[Beasley et Jörnsten, 1992] Beasley, J. et Jörnsten, K. (1992). Enhancing an algorithm for set covering problems. *European Journal of Operational Research*, 58:293–300.

[Berg *et al.*, 2008] Berg, M., Cheong, O., Kreveld, M. et Overmars, M. (2008). *Computational Geometry : Algorithms and Applications*. Springer.

[Berge, 1973] Berge, C. (1973). *Graphes et hypergraphes*. Dunod.

[Berman *et al.*, 1989] Berman, O., Drezner, Z., Tamir, A. et Wesolowsky, G. (1989). An overview of representative problems in location research. *Management Science*, 35: 645–674.

[Berman *et al.*, 2009] Berman, O., Drezner, Z., Tamir, A. et Wesolowsky, G. (2009). Optimal location with equitable loads. *Annals of operations research*, 167:307–325.

[Bichot et Siarry, 2011] Bichot, C.-E. et Siarry, P. (2011). *Graph Partitioning*. John Wiley and Sons.

[Bozkaya *et al.*, 2003] Bozkaya, B., Erkut, E. et Laporte, G. (2003). A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research*, 144(1):12–26.

[Bozkaya *et al.*, 2005] Bozkaya, B., Erkut, E. et Laporte, G. (2005). *Political Districting : Solving a Multi-Objective Problem Using Tabu Search*. Springer.

[Brandeau et Chiu, 1989] Brandeau, M. et Chiu, S. (1989). An overview of representative problems in location research. *Management Science*, 35:645–674.

[Brunetta *et al.*, 1997] Brunetta, L., Conforti, M. et Rinaldi, G. (1997). A branch-and-cut algorithm for the equicut problem. *Mathematical Programming*, 78:243–263.

[Caprara *et al.*, 1999] Caprara, A., Fischetti, M. et Toth, P. (1999). A heuristic method for the set covering problem. *Operations Research*, 47:730–743.

[Caprara *et al.*, 2000] Caprara, A., Fischetti, M. et Toth, P. (2000). Algorithms for the set covering problem. *Annals of Operations Research*, 98:353–371.

[Carrizosa *et al.*, 1995] Carrizosa, E., Conde, E., Munoz, M. et Puerto, J. (1995). The generalized weber problem with expected distances. *RAIRO*, 29:35–57.

[Ceselli, 2003] Ceselli, A. (2003). Two exact algorithms for the capacitated p-median problem. *4OR : Quarterly Journal of the Belgian, French and Italian Operations Research Society*, 1:319–340.

[Charikar *et al.*, 1999] Charikar, M., Chekuri, C., Cheung, T., Dai, Z., Goel, A., Guha, S. et Li, M. (1999). Approximation algorithms for directed steiner problems. *Journal of Algorithms*, 33:73–91.

[Christofides et Beasley, 1982] Christofides, N. et Beasley, J. (1982). A tree search algorithm for the p-median problem. *European Journal of Operational Research*, 10(2): 196–204.

[Church et Meadows, 1979] Church, R. et Meadows, M. (1979). Location modeling utilizing maximum service distance criteria. *Geographical Analysis*, 11:358–373.

[Church et ReVelle, 1974] Church, R. et ReVelle, C. (1974). The maximal covering location problem. *Papers of the Regional Science Association*, 32:101–118.

[Church, 1974] Church, R. L. (1974). *Synthesis of a Class of Public Facilities Location Models*. Thèse de doctorat, University Microfilms.

[Collette et Siarry, 2002] Collette, Y. et Siarry, P. (2002). *Optimisation multiobjectif*. Eyrolles.

[Daskin, 1995] Daskin, M. (1995). *Network and Discrete Location : Models, Algorithms and Applications*. JohnWiley.

[Donath et Hoffman, 1973] Donath, W. et Hoffman, A. (1973). Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17, Number 5:420–425.

[Downs et Camm, 1996] Downs, B. et Camm, J. (1996). An exact algorithm for the maximal covering location problem. *Naval Research Logistics Quarterly*, 43:435–461.

[Dreyfus et Wagner, 1972] Dreyfus, S. et Wagner, R. (1972). The steiner problem in graphs. *Networks*, 1:195–207.

[Drezner et Drezner, 2006] Drezner, T. et Drezner, Z. (2006). Multiple facilities location in the plane using the gravity model. *Geographical Analysis*, 38:391–406.

[Driebeek, 1966] Driebeek, N. (1966). An algorithm for the solution of mixed integer programming problems. *Management Science*, 12:576–587.

[Dwyer et Evans, 1981] Dwyer, F. et Evans, J. (1981). A branch and bound algorithm for the list selection problem in direct mail advertising. *Management Science*, 27:658–667.

[Dyer et Frieze, 1985] Dyer, M. et Frieze, A. (1985). A simple heuristic for the p-center problem. *Operations Research Letters*, 3:285–288.

[Ehrgott, 2005] Ehrgott, M. (2005). *Multicriteria Optimization, 2nd Edition.* Springer.

[Eiselt *et al.*, 1993] Eiselt, H., Laporte, G. et Thisse, J.-F. (1993). Competitive location models : A framework and bibliography. *Transportation Science*, 27(1):44–54.

[Eiselt et Sandblom, 2004] Eiselt, H. et Sandblom, C.-L. (2004). *Decision Analysis, Location Models, and Scheduling Problems.* Springer-Verlag, Berlin.

[Erkut et Newman, 1989] Erkut, E. et Newman, S. (1989). Analytical models for locating undesirable facilities. *European Journal of Operational Research*, 40:275–291.

[Fleischmann et Paraschis, 1988] Fleischmann, B. et Paraschis, J. (1988). Solving a large scale districting problem : A case report. *Computers and Operations Research*, 15:521–533.

[Galvao *et al.*, 2000] Galvao, R., Espejo, L. et Boffey, B. (2000). A comparison of lagrangean and surrogate relaxations for the maximal covering location problem. *European Journal of Operational Research*, 124:377–389.

[Galvao et ReVelle, 1996] Galvao, R. et ReVelle, C. (1996). A lagrangean heuristic for the maximal covering location problem. *European Journal of Operational Research*, 88:114–123.

[Garey *et al.*, 1976] Garey, M., Johnson, D. et Stockmeyer, L. (1976). Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267.

[Garey et Johnson, 1979] Garey, M. R. et Johnson, D. S. (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company.

[Garfinkel et Nemhauser, 1970] Garfinkel, R. et Nemhauser, G. (1970). Optimal political districting by implicit enumeration techniques. *Management Science*, 16:495–508.

[George *et al.*, 1997] George, J., Lamar, B. et Wallace, C. (1997). Political district determination using large-scale network optimization. *Socio-Economic Planning Sciences*, 31:11–28.

[Glover et Laguna, 1997] Glover, F. et Laguna, M. (1997). *Tabu search.* Kluwer Academic.

[Golden et Skiscim, 1986] Golden, B. et Skiscim, C. (1986). Using simulated annealing to solve routing and location problems. *Naval Research Logistic Quarterly*, 33:261–279.

[Grilli di Cortona *et al.*, 1999] Grilli di Cortona, P., Manzi, C., Pennisi, A., Ricca, F. et Simeone, B. (1999). *Evaluation and optimization of electoral systems.* Society for Industrial and Applied Mathematics.

[Guibas *et al.*, 1992] Guibas, L., Knuth, D. et Sharir, M. (1992). Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413.

[Hakimi, 1964] Hakimi, S. (1964). Optimum location of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12:450–459.

[Hakimi, 1965] Hakimi, S. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, 13(3):462–475.

[Hakimi, 1971] Hakimi, S. (1971). Steiner's problem in graphs and its implications. *Networks*, 1:113–133.

[Hamacher *et al.*, 1998] Hamacher, H., Labbé, M. et Nickel, S. (1998). Multicriteria network location problems with sum objective. *Networks*, 33:79–92.

[Hamacher et Nickel, 1996] Hamacher, H. et Nickel, S. (1996). Multicriteria planar location problems. *European Journal of Operational Research*, 94:66–86.

[Hamacher et Nickel, 1998] Hamacher, H. et Nickel, S. (1998). Classification of location models. *Location Science*, 6:229–242.

[Hamacher *et al.*, 2007] Hamacher, H., Pedersen, C. et Ruzika, S. (2007). Finding representative systems for discrete bicriterion optimization problems. *Operations Research Letters*, 35:336–344.

[Handler et Mirchandani, 1979] Handler, G. et Mirchandani, P. (1979). *Location on Networks : Theory and Algorithms*. MIT Press, Cambridge.

[Hansen et Jaumard, 1997] Hansen, P. et Jaumard, B. (1997). Cluster analysis and mathematical programming. *Mathematical Programming*, 79:191–215.

[Hendrickson et Leland, 1995] Hendrickson, B. et Leland, R. (1995). An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing*, 16(2):452–469.

[Hess *et al.*, 1965] Hess, S., F, Weaver, J., Siegfeldt, J., Whelan, J. et Zhitlau, P. (1965). Nonpartisan political districting by computer. *Operations Research*, 13:998–1006.

[Hochbaum et Shmoys, 1985] Hochbaum, D. et Shmoys, D. (1985). A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10:180–184.

[Hojati, 1996] Hojati, M. (1996). Optimal political districting. *Computers and Operations Research*, 23:1147–1161.

[Horowitz et Sahni, 1976] Horowitz, E. et Sahni, S. (1976). Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, 23:317–327.

[Ilhan et Pinar, 2001] Ilhan, T. et Pinar, M. (2001). An efficient exact algorithm for the vertex p-center problem. *http ://www.optimization-online.org/DB-HTML/2001/09/376.html*.

[Johnson *et al.*, 1993] Johnson, E., Mehrotra, A. et Nemhauser, G. (1993). Min-cut clustering. *Mathematical Programming*, 62:133–151.

[Järvinen *et al.*, 1972] Järvinen, P., Rajala, J. et Sinervo, H. (1972). A branch-and-bound algorithm for seeking the pmedian. *Operations Research*, 20(1):173–178.

[Kalcsics *et al.*, 2010] Kalcsics, J., Melo, T., Nickel, S. et Rodríguez-Chía, A. (2010). The ordered capacitated facility location problem. *TOP : An Official Journal of the Spanish Society of Statistics and Operations Research*, 18:203–222.

[Kalcsics *et al.*, 2002] Kalcsics, J., Nickel, S. et Puerto, J. a. (2002). Planning sales territories - a facility location approach. *in Operations Research Proceedings 2001, Springer Verlag Berlin*, pages 141–148.

[Kariv et Hakimi, 1979a] Kariv, O. et Hakimi, S. (1979a). An algorithmic approach to network location problems, part I : the p-centers. *SIAM Journal of Applied Mathematics*, 37:513–538.

[Kariv et Hakimi, 1979b] Kariv, O. et Hakimi, S. (1979b). An algorithmic approach to network location problems part I : the p-medians. *SIAM Journal of Applied Mathematics*, 37:539–560.

[Karp, 1972] Karp, R. (1972). Reducibility among combinatorial problems. *Complexity of Computer Computations, Plenum, New York*, pages 85–103.

[Karypis et Kumar, 1998] Karypis, G. et Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal of Scientific Computing*, 20(1):359–392.

[Kernighan et Lin, 1970] Kernighan, B. et Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307.

[Kuehn et Hamburger, 1963] Kuehn, A. et Hamburger, M. (1963). A heuristic program for locating warehouses. *Management Science*, 9(4):643–666.

[Lorena et Lopes, 1994] Lorena, L. et Lopes, F. (1994). A surrogate heuristic for set covering problems. *European Journal of Operational Research*, 79:138–150.

[Maniezzo *et al.*, 1998] Maniezzo, V., Mingozzi, A. et Baldacci, R. (1998). A bionomic approach to the capacitated p-median problem. *Journal of Heuristics*, 4:263–280.

[Maranzana, 1964] Maranzana, F. (1964). On the location of supply points to minimize transport costs. *Operations Research Quarterly*, 15(3):261–270.

[Martinich, 1988] Martinich, J. (1988). A vertex-closing approach to the p-center problem. *Naval Research Logistics*, 35:185–201.

[Megiddo *et al.*, 1983] Megiddo, N., Zemel, E. et Hakimi, S. (1983). The maximal coverage location problem. *SIAM Journal of Algebraic and Discrete Methods*, 4:253–261.

[Mehrotra *et al.*, 1998] Mehrotra, A., Johnson, E. et Nemhauser, G. (1998). An optimization based heuristic for political districting. *Management Science*, 44:1100–1114.

[Miller *et al.*, 1960] Miller, C., Tucker, A. et Zemlin, R. (1960). Integer programming formulation of the travelling salesman problem. *Journal of the Association for Computing Machinery*, 7:326 –329.

[Minieka, 1970] Minieka, E. (1970). The m-center problem. *SIAM*, 12:138–139.

[Mitchell, 2001] Mitchell, J. (2001). Branch-and-cut for the k-way equipartition problem. Rapport technique, Mathematical Sciences, Rensselaer Polytechnic Institute.

[Mladenović *et al.*, 2003] Mladenović, N., Labbé, M. et Hansen, P. (2003). Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, 42:48–64.

[Mulvey et Beck, 1984] Mulvey, J. et Beck, P. (1984). Solving capacitated clustering problems. *European Journal of Operational Research*, 18:339–348.

[Nemhauser et Wolsey, 1999] Nemhauser, G. et Wolsey, L. (1999). *Integer and Combinatorial Optimization*. Wiley-Interscience, New York.

[Niedermeier, 2008] Niedermeier, R. (2008). *Invitation to Fixed-Parameter Algorithms, Oxford lecture series in mathematics and its application 31.* Oxford University Press, USA.

[Osman et Christofides, 1994] Osman, I. et Christofides, N. (1994). Capacitated clustering problems by hybrid simulated annealing and tabu search. *International Transactions in Operational Research*, 13:317–336.

[Osorio *et al.*, 2002] Osorio, M., Glover, F. et Hammer, P. (2002). Cutting and surrogate constraint analysis for improved multidimensional knapsack solutions. *Annals of Operations Research*, 117:71–93.

[Plastria, 1995] Plastria, F. (1995). Facility location : A survey of applications and methods. *In Drezner, Z. (Ed.) : Continuous location problems*, Springer-Verlag, Berlin: 225–262.

[Pothen *et al.*, 1990] Pothen, A., Simon, H. et Liou, K. (1990). Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452.

[Puerto et Fernandez, 1999] Puerto, J. et Fernandez, F. (1999). Multicriteria mini-sum facility location problems. *Journal of the Multicriteria Decision Analysis*, 8:268–280.

[Ricca et Simeone, 2008] Ricca, F. et Simeone, B. (2008). Local search algorithms for political districting. *European Journal of Operational Research*, 189(3):1409–1426.

[Ross et Soland, 1977] Ross, G. et Soland, R. (1977). Modeling facility location problems as generalized assignment problems. *Management Science*, 24:345–357.

[Savelsberg, 1994] Savelsberg, M. (1994). Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal of Computing*, 6:445–454.

[Schilling *et al.*, 1993] Schilling, D. A., Vaidyanathan, J. et Barkhi, R. (1993). A review of covering problems in facility location. *Location Science*, 1:25–55.

[Sensen, 2001] Sensen, N. (2001). Lower bounds and exact algorithms for the graph partitioning problem using multi-commodity flows. *Lecture notes in computer science*, 2161:391–403.

[Simon, 1991] Simon, H. (1991). Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2:135–148.

[Tang *et al.*, 2010a] Tang, X., Soukhal, A. et T'kindt, V. (2010a). Le problème de la sectorisation à partir de centres en cartographie : résolution par la recherche opérationnelle. *ROADEF2010, Toulouse, France.*

[Tang *et al.*, 2010b] Tang, X., Soukhal, A. et T'kindt, V. (2010b). Sampling the pareto front for a bicriteria sectorization problem. *The 9th International Conference on Multiple Objective Programming and Goal Programming, Sousse, Tunisie.*

[Tang *et al.*, 2011a] Tang, X., Soukhal, A. et T'kindt, V. (2011a). A branch and bound algorithm for a bicriteria sectorization problem. *OR 2011, Zurich, Suisse.*

[Tang *et al.*, 2011b] Tang, X., Soukhal, A. et T'kindt, V. (2011b). Preprocessing for a map sectorization problem by means of mathematical programming. *Annals of Operations Research, en cours de révision.*

[Tang *et al.*, 2011c] Tang, X., Soukhal, A. et T'kindt, V. (2011c). Le problème de la sectorisation multicritère en cartographie : approche par les modèles de localisation-allocation. *ROADEF2011, Saint-Étienne, France.*

[Tang *et al.*, 2012a] Tang, X., Soukhal, A. et T'kindt, V. (2012a). Preprocessing for a map sectorization problem by means of mathematical programming. *ORP3, Linz, Autriche.*

[Tang *et al.*, 2012b] Tang, X., Soukhal, A. et T'kindt, V. (2012b). Le problème de la sectorisation multicritère en cartographie : approche par programmation linéaire. *ROADEF2012, Angers, France.*

[Tarjan, 1972] Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal Computing*, 1:146–160.

[Teitz et Bart, 1968] Teitz, M. et Bart, P. (1968). Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research*, 16(5):955–961.

[Tercinet, 2004] Tercinet, F. (2004). *Méthodes arborescentes pour la résolution des problèmes d'ordonnancement, conception d'un outil d'aide au développement.* Thèse de doctorat, Polytech'Tours, Université François Rabelais Tours.

[T'kindt et Billaut, 2006] T'kindt, V. et Billaut, J.-C. (2006). *Multicriteria Scheduling : Theory, Models and Algorithms, 2nd Edition.* Springer.

[T'kindt *et al.*, 2007] T'kindt, V., Della Croce, F. et Bouquard, J.-L. (2007). Enumeration of pareto optima for a flowshop scheduling problem with two criteria. *INFORMS Journal on Computing*, 19(1):64–72.

[T'kindt *et al.*, 2004] T'kindt, V., Della Croce, F. et Esswin, C. (2004). Revisiting branch and bound search strategies for machine scheduling problems. *Journal of Scheduling*, 7(6):429–440.

[Tomlin, 1971] Tomlin, J. (1971). An improved branch-and-bound method for integer programming. *Operations Research*, 19:1070–1075.

Résumé :

Les travaux présentés dans cette thèse visent à proposer des méthodes pour résoudre les problèmes de la sectorisation multicritère en cartographie. En premier temps, nous avons défini les problèmes différents de la sectorisation et nous avons établi les liens entre ces problèmes avec les problèmes classiques qui sont bien étudiés dans la littérature : le problème de découpage de district politique, les problèmes de localisation et le problème du partitionnement de graphe.

Deux types de méthodes ont été abordés pour résoudre les problèmes de sectorisation. Des heuristiques ont été développées et elles consistent à calculer un optimum de Pareto pour les différents problèmes. Et pour le problème de sectorisation à partir de pôles, nous avons aussi utilisé et expérimenté un algorithme de boîte pour trouver une représentation du front de pareto.

La méthode exacte *branch and bound* a été utilisée pour résoudre le problème de sectorisation sans pôle prédéfini optimalement. Avant que nous appliquons cette procédure, nous ajoutons quelques inégalités valides dans la formulation mathématique pour restreindre l'espace des solutions et nous développons une procédure de prétraitement pour réduire la taille du problème.

Mots clés :

sectorisation cartographique, multicritère, prétraitement

Abstract :

The work presented in this thesis aims to propose methods to solve the multicriteria map sectorization problem in cartography. Firstlly, we have defined the different sectorization problems and we have established the links between these problems with some classical problems which are well studied in the literature : political districting problem, location-allocation problems and constrained graph partitioning problems.

Two types of methods have been proposed to solve the sectorization problem. Heuristics have been developed and they compute an optimum Pareto for the different sectorization problems. And for the sectorization problem with predefined centers, we have used a box algorithm and experimented it to find a representation of the Pareto front.

The branch and bound method was used to solve optimally the sectorization problem without predefined centers. Before we apply this procedure, we add some valid inequalities in the mathematical formulation for restrict the space of solutions and we develop a preprocessing procedure to reduce the size of the problem.

Keywords :

cartography sectorization, multicriteria, preprocessing