Titre: Title:	Partitionnement d'une zone géographique en territoires homogènes et contigus
Auteur: Author:	Pierre De la Poix de Fréminville
Date:	2012
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	De la Poix de Fréminville, P. (2012). Partitionnement d'une zone géographique en territoires homogènes et contigus [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/832/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/832/
Directeurs de recherche: Advisors:	Louis-Martin Rousseau, & Guy Desaulniers
Programme: Program:	Mathématiques appliquées

UNIVERSITÉ DE MONTRÉAL

PARTITIONNEMENT D'UNE ZONE GÉOGRAPHIQUE EN TERRITOIRES HOMOGÈNES ET CONTIGUS

PIERRE DE LA POIX DE FRÉMINVILLE
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION DU DIPLÔME DE
MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
AVRIL 2012

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

PARTITIONNEMENT D'UNE ZONE GÉOGRAPHIQUE EN TERRITOIRES HOMOGÈNES ET CONTIGUS

présenté par : M. DE LA POIX DE FRÉMINVILLE Pierre
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury constitué de :

Mme LAHRICHI Nadia, Ph.D., présidente
M. ROUSSEAU Louis-Martin, Ph.D., membre et directeur de recherche
M. DESAULNIERS Guy, Ph.D., membre et directeur de recherche
M. PERRON Sylvain, Ph.D., membre

à tous mes amis de Montréal,

Remerciements

Je tiens en premier lieu à remercier mes directeurs de recherche Louis-Martin Rousseau et Guy Desaulniers, pour avoir accepté de diriger ma maîtrise et pour leur aide régulière. Je remercie également l'École Polytechnique de Montréal ainsi que la compagnie pour laquelle ce projet à été réalisé, qui ont participé au financement de mes études.

Je remercie François Guertin, Vincent Boyer pour leur aide sur la bibliothèque OOBB.

Je remercie Serge Bisaillon pour ses conseils sur l'implémentation de mon programme.

Je remercie aussi mon tuteur dans la compagnie ainsi que son associé pour leur grande disponibilité, leur implication dans le projet, ainsi que pour l'accueil très chaleureux qui m'a été réservé.

Je remercie mes amis de Montréal, du GERAD et du CIRRELT, grâce à qui j'ai passé une année et demi très enrichissante.

Je remercie enfin tous ceux que j'aurais pu oublier et qui ont contribué de près ou de loin à cette maîtrise, en particulier tous les professeurs que j'ai pu avoir et qui m'ont permis d'aller toujours plus loin dans mes études.

Résumé

Le problème de régionalisation ou de "districting" consiste à diviser une zone géographique en un nombre prédéfini de sous-zones contigües tout en minimisant un critère de partitionnement fonction de données non géographiques. Le problème de régionalisation peut être vu comme un processus de regroupement d'unités élémentaires, les unités géographiques (UG), en groupes appelés territoires qui, une fois assemblés, reconstituent la carte ou la configuration donnée. Ce problème a surtout été étudié dans le cadre du découpage électoral.

Le problème qui nous intéresse consiste à grouper les UG selon une valeur associée à chacune d'elle en des territoires homogènes respectant un poids minimum. Pour cela nous utilisons comme critère d'agrégation la variance intra-territoire qui est la somme pondérée de la variance de chaque territoire en solution. La variance d'un territoire est la variance de la valeur associée à chaque UG lui appartenant, et la pondération d'un territoire dans l'objectif est la somme des poids de chaque UG qui lui est associé.

Ce problème est difficile à résoudre et une technique d'énumération de tous les territoires n'est donc pas envisageable pour de grandes instances. La difficulté par rapport à quelques travaux déjà réalisés est la présence simultanée d'une fonction objectif quadratique et d'une contrainte de contiguïté, ainsi que la taille des instances (500 UG).

Ce travail présente une méthode heuristique de génération de colonnes couplée avec une méthode de branchement pour résoudre un tel problème de partitionnement avec contrainte de contiguïté. Dans la méthode de génération de colonnes, le sous-problème génère de nouveaux territoires et est résolu par un algorithme heuristique de type glouton avec plusieurs points de départ. La méthode de branchement est aussi heuristique car les décisions prises sont fixées de façon permanente, i.e., aucun retour en arrière n'est permis dans l'arbre de branchement. A notre connaissance une telle méthode de résolution avec des instances de l'ordre de 500 UG n'a pas encore été appliquée. Cette méthode a été développée dans un contexte industriel et permet d'obtenir des solutions réalisables de bonne qualité sur les instances testées dans des temps relativement courts (15 min. à 40 min.).

Abstract

The regionalization or districting problem consists of dividing a geographical area into a predefined number of contiguous territories, while optimizing a clustering criterion. The regionalization problem can be seen as a process of aggregating elementary geographical units (GU) into clusters called territories, that combined, cover the entire map or given configuration. The most studied variant of this problem is the electoral districting problem.

The variant of the regionalization problem studied here, referred to as the PPHCT, consists of aggregating the GU according to their value into homogeneous and contiguous territories, satisfying a minimum weight constraint. For this matter, an aggregation criterion, namely the within-territory variance, is used, which is the weighted sum of the variance of each territory in the solution. The variance of a territory is the variance of the value assigned to each GU in that territory, and the weight of a territory is the sum of the weights of each GU in that territory.

The PPHCT is difficult to solve optimally and an enumeration of all the feasible territories cannot be applied for large instances. The main difficulty of this variant, in comparison to other variants previously studied, is the simultaneous presence of a contiguity constraint and a quadratic objective function, together with large instances (500 GU).

The purpose of this paper is to present a heuristic column-generation model and branch-and-bound algorithm designed to solve the PPHCT. In the column-generation method, the sub-problem generates new feasible territories and is solved by a greedy multi-start algorithm. The branching method is also heuristic, as branching decisions are taken permanently, that is, no back-tracking is possible in the branching tree. This solution method was developed in an industrial context and is able to produce good quality feasible solutions on the tested data, in relatively short computing times (15 min. to 40 min.).

Table des matières

Liste des tableaux

Liste des figures

Liste des sigles et abréviations

MIP	problème d'optimisation linéaire en nombre entiers
PDE	problème de découpage électoral
PL	problème linéaire
PMR	problème maître restreint
PPTHC	problème de partitionnement en territoires homogènes et contigus
UG	unité géographique

Chapitre 1

INTRODUCTION

1.1 Le problème de régionalisation

Le problème de régionalisation consiste à diviser une zone géographique en un nombre prédéfini de sous-zones contigües tout en minimisant un critère de partitionnement fonction de données non-géographiques. On le retrouve aussi dans la littérature sous le nom de problème de partitionnement sous contrainte, de découpage de zone, ou encore de "districting". Ce problème a surtout été étudié dans le cadre du découpage électoral (Ricca et Simeone (2008)).

Des variantes du problème ont aussi été étudiées dans des domaines variés, par exemple : dans le cadre du découpage de zones pour la police (D'Amico *et al.* (2002)), du découpage scolaire (Ferland et Guénette (1990)), du découpage pour les opérations d'épandage de sel (Muyldermans *et al.* (2002)), mais aussi pour établir des zones de travail pour une équipe de voyageurs de commerce (Zoltners et Sinha (1983)), des zones où installer les centres des réseaux internet métropolitains (Park *et al.* (2000)), des zones d'énergie électrique (Bergey *et al.* (2003)), des zones pour les patients d'une clinique publique (Gascon *et al.* (2010)), des zones de soin à domicile (Blais *et al.* (2003)) et enfin des zones dans le cadre de réseaux de transport (Tavares-Pereira *et al.* (2007)).

Ce problème fait partie de la classe générale des problèmes de partitionnement (ou de "clustering") qui consiste à regrouper un ensemble d'unités en un nombre prédéfini de classes ou "clusters" tout en minimisant un critère d'agrégation. Le problème de régionalisation peut être vu comme un processus de regroupement d'unités élémentaires qu'on appellera par la suite unités géographiques (UG), en groupes appelés territoires, qui, une fois assemblés, reconstituent la carte ou la configuration donnée.

Il existe plusieurs manières de regrouper les UG, et donc de définir le critère de partitionnement. Si on veut grouper des objets similaires en des groupes homogènes, une manière possible de procéder est d'utiliser comme critère d'agrégation la variance

intra-classe qui est la somme pondérée des variances de chaque groupe. La variance d'un groupe est la variance de la valeur associée à chaque objet au sein du groupe, et la pondération d'un groupe est la somme des poids de chaque objet associé à ce groupe. Minimiser ce critère revient aussi à maximiser la variance inter-classes qui est la somme pondérée des écarts au carré de la différence entre la moyenne de chaque groupe et la moyenne totale (Apostol et Mnatsakanian (2003)).

Certaines variantes du problème ont été prouvées NP-complètes (Altman (1997); Aloise (2009)). Ce problème est donc a priori difficile à résoudre et une technique d'énumération n'est donc pas envisageable pour de grandes instances. De plus les contraintes de contiguïté dans un modèle de découpage de carte sont difficiles à prendre en compte et dans certaines formulations du problème, celles-ci peuvent engendrer un nombre exponentiel de contraintes (en fonction du nombre d'UG).

Ce travail présente une méthode heuristique de génération de colonnes pour résoudre un tel problème de partitionnement avec contraintes de contiguïté, qui a pour objectif la minimisation de la variance intra-territoire. La difficulté de ce problème par rapport à quelques travaux déjà réalisés est la présence simultanée d'un objectif quadratique et de contraintes de contiguïté.

1.2 Définitions et concepts de base

Considérons une carte géographique donnée qui est divisée en un ensemble U d'unités géographiques (UG).

A chaque UG $u \in U$ on associe :

- une valeur v_u,
- un poids ω_u,
- des coordonnées géographiques (position du centroïde),
- la liste des UG voisines de u.

Le problème qui nous intéresse, que l'on appelera PPTHC (problème de partitionnement en territoires homogènes et contigus), consiste à partitionner cette carte en un nombre fixé N de territoires. Un territoire regroupe un ensemble d'UG contigües. Il est réalisable si la somme des poids de ces UG est plus grande ou égale à un poids minimal. Chaque UG appartient à exactement un territoire.

Une partition est donc réalisable si :

- elle comporte N territoires réalisables,

– et si chaque UG est assignée à un seul territoire.

Chaque territoire est réalisable si ce territoire :
– est un ensemble contigu d'UG,
– et a un poids minimal ω_{min}.

Parmi toutes les partitions réalisables, on cherche une partition qui minimise la variance intra-territoire Var_{intra}, qui est une somme pondérée des variances de chaque territoire. Dénotons par :
– $\Omega_t = \sum_{u \in t} \omega_u$: le poids d'un territoire t.
– $\mu_t = \frac{\sum_{u \in t} \omega_u v_u}{\Omega_t}$: la moyenne d'un territoire.
– $Var_t = \frac{\sum_{u \in t} \omega_u (v_u - \mu_t)^2}{\Omega_t}$: la variance d'un territoire.

En numérotant de 1 à N les territoires sélectionnés, la variance intra-territoire s'exprime alors (le dénominateur est une constante) :

$$Var_{intra} = \frac{\sum_{t=1}^{N} \Omega_t . Var_t}{\sum_{j=1}^{N} \Omega_j}$$

A cela on peut rajouter des contraintes de sous-zones : c'est-à-dire qu'un sous-ensemble S d'UG de la carte peut être couvert au maximum par un nombre N^S de territoires.

1.3 Objectifs de recherche

Ce projet de maîtrise a été réalisé dans un contexte industriel, qui ne peut être dévoilé par souci de confidentialité. Dans ce contexte, le problème de régionalisation considéré se pose pour de nombreuses zones géographiques (au moins une dizaine) et est résolu actuellement une fois tous les quatre ou cinq ans pour chacune de ces zones. La résolution étant faite en grande partie manuellement, elle requiert un certain temps et mène à des solutions de qualité incertaine. L'objectif de recherche principal consiste à développer une méthode d'optimisation efficace pour résoudre ce problème à partir d'une solution initiale correspondant à la solution actuellement mise en place. Le but n'est donc pas de développer une méthode exacte mais plutôt une méthode appliquée qui produit des résultats rapidement afin de pouvoir analyser un grand nombre de

scénarios en peu de temps.

1.4 Plan du mémoire

Ce mémoire est structuré comme suit. Après avoir fait un état de l'art des méthodes de régionalisation présentées dans la littérature au chapitre 2, nous présentons au chapitre 3 le modèle mathématique adopté ainsi que l'algorithme de résolution qui a été développé. Après avoir donné quelques détails sur notre implantation au chapitre 4, nous présentons les résultats obtenus au chapitre 5. Enfin, une brève conclusion est proposée au chapitre 6.

Chapitre 2

REVUE DE LITTÉRATURE

L'analyse statistique de données spatiales requiert souvent l'agrégation d'unités géographiques (UG) basiques en unités plus grandes, appelées régions ou territoires, afin par exemple de préserver la confidentialité, de minimiser les différences de population, de réduire certaines imprécisions des données, ou encore de simplifier la visualisation et l'interprétation de l'information d'une carte. Les régions créées en utilisant des critères géographiques et/ou des critères socio-économiques (ex : homogénéité, complémentarité, économies régionales, etc.), sont appelées des régions analytiques.

L'article de Duque *et al.* (2007) résume 40 ans de recherche sur le sujet des méthodes de régionalisation. Des synthèses similaires, antérieures à cet article, ont été réalisées par Murtagh (1985) et Gordon (1996, 1999). Les méthodes ont toutes en point commun les caractéristiques suivantes :

- elles agrègent des UG en un nombre prédéfini de régions, en optimisant des critères d'agrégation particuliers ;
- les régions doivent être contigües (toutes les UG connectées géographiquement) ;
- chaque UG doit être assignée à une et une seule région.

Dans ce qui suit, nous allons étudier dans un premier temps des résultats concernant la minimisation de fonctions objectif semblables à celle qui nous intéresse, c'est-à dire la minimisation de la somme des carrés. Puis dans un deuxième temps, nous étudierons des méthodes de résolution pour des variantes du problème de régionalisation. En nous inspirant de Duque *et al.* (2007), nous classerons ces articles en plusieurs groupes :

- sans contrainte explicite de contiguïté ;
- avec contrainte explicite de contiguïté :
  - méthodes d'optimisation exactes,
  - méthode heuristiques,
  - méthode heuristiques mixtes.

2.1 La minimisation de la somme des carrés

Le problème de minimisation de la somme des carrés a été résolu de manière exacte dans la thèse de Aloise (2009). Le problème étudié est un problème de classification automatique : étant donné n points $p_i | i \in 1, \ldots, n$ de l'espace euclidien, on cherche à réaliser une partition de ces points en k classes, telles que la somme des carrés des distances de chaque point au centroïde y_j de sa classe est minimisée ($n >> k$).

L'expression mathématique du problème est la suivante :

$$\min_{x,y} \sum_{i=1}^{n} \sum_{j=1}^{k} x_{ij} ||p_i - y_j||^2$$

Les paramètres du modèle sont :

– n : nombre de points.

– k : nombre de classes.

– p_i : position du point i.

Les variables du modèle sont :

– y_j : position du centroïde de la classe j.

– $x_{ij} = 1$ si le point i appartient à la classe j, 0 sinon.

Il convient de noter que les conditions de premier ordre sur le gradient de la fonction objectif imposent que les centres des classes optimales soient aux centroïdes des classes.

Aloise présente une méthode de génération de colonnes inspirée de celle de Merle *et al.* (2000) avec une nouvelle manière de résoudre le problème auxiliaire, basée sur des arguments géométriques. Il utilise la règle de Ryan et Foster (1981) quand il est nécessaire de brancher. Cette méthode exacte permet de résoudre des problèmes de grande taille (jusqu'à $n = 2000$ entités et un ratio n/k ne s'éloignant pas trop de 10).

Ce problème est donc similaire au nôtre. Cependant il n'y a pas de contrainte de contiguïté.

2.2 Les méthodes de régionalisation

2.2.1 Sans contrainte explicite de contiguïté

Méthode conventionnelle de partitionnement

Cette méthode en deux étapes est la plus simple. Elle consiste à utiliser un algorithme classique de partitionnement, puis à effectuer un traitement des territoires obtenus pour les rendre contigus. Cette méthode a comme qualité d'induire de l'homogénéité lors de la première étape. Cependant, les régions obtenues ne sont pas forcément compactes. La forme dépend essentiellement de la distribution des variables et de l'algorithme de partitionnement choisi pour l'analyse.

On peut citer la méthode en deux étapes proposée par Openshaw et Wymer (1995). La première étape consiste à agréger les unités en termes de variables (méthode conventionnelle ou hiérarchique). La deuxième consiste à diviser les régions obtenues par composante connexe. On doit ajuster le nombre de classes pour obtenir le nombre de régions désiré.

Les algorithmes conventionnels de partitionnement comme l'algorithme des k-moyennes (MacQueen (1967)) autorisent seulement les mouvements qui améliorent la solution courante. L'algorithme converge rapidement mais peut être bloqué dans un minimum local. La solution finale est de plus très dépendante de la position initiale des centroïdes. Une approche possible consiste à partir de différents ensembles de centroïdes initiaux, puis sélectionner la meilleure solution. Une variante utilisant le recuit simulé a aussi été proposée par Openshaw et Wymer (1995). Elle autorise les mouvements qui n'améliorent pas la solution.

Une variante utilisant une méthode de recherche locale, appelée méthode des j-moyennes, a aussi été proposée par P. Hansen (2001). Elle considère toutes les réallocations de centres de territoires possibles.

Régionalisation par maximisation de la compacité des régions

Une façon d'obtenir la contiguïté est de forcer les régions à être les plus compactes possibles. Cette méthode consiste la plupart du temps à définir des centres de région et à assigner des UG aux centres tout en minimisant un critère d'agrégation pour que les territoires créés soient les plus compacts possibles. Le plus souvent les auteurs proposent de rapprocher la forme des régions de celle d'un carré ou d'un cercle (Wise

et al. (1997)). Le bon choix des centres de région est une étape importante du procédé, c'est pourquoi par exemple, Bacao *et al.* (2005) proposent une méthodologie qui applique les algorithmes génétiques pour définir les centres de région. Cependant, ces méthodes de compacité ne laissent pas vraiment de liberté sur la forme des territoires, et la solution finale dépend fortement du choix des centroïdes.

La méthode de régionalisation par maximisation de la compacité a été introduite par Weaver et Hess (1963) pour le problème de découpage électoral (PDE). Ce problème consiste à redessiner les frontières politiques entre districts, suite à des modifications de population ou du nombre de sièges de représentant politique. Le PDE a une contrainte supplémentaire qui veut que les populations des différents territoires créés soient égales. Etant donné une zone géographique, et ses subdivisions (UG), on veut diviser cette zone en m districts le plus compacts possibles tels que chaque district ait la même taille de population de votants, avec parfois une certaine tolérance (Hess *et al.* (1965)).

Une manière de procéder est décrite dans un des premiers articles sur le PDE, soit Hess *et al.* (1965). Ils utilisent un modèle itératif de problème de location-allocation, et formulent le problème sous forme d'un programme en nombres entiers qui consiste à assigner les UG à des centres définis dans les paramètres. L'objectif est de maximiser la compacité en minimisant les moments d'inertie (la somme des carrés des distances au centroïde du territoire multipliée par la population). La formulation requiert aussi des populations égales dans les régions. Pour cela, ils permettent des assignations fractionnaires, assignent chaque morceau d'UG au centre qui fournit la plus grande fraction de sa demande et recalculent les centres de région. Le processus (placement des centres, puis résolution) est répeté jusqu'à ce qu'un critère de convergence soit satisfait. Enfin, une inspection de la solution finale pour la contiguïté est réalisée.

Une autre manière de procéder est d'utiliser comme critère une fonction multi-objectifs (homogénéité, compacité, équité des populations). Cette fonction est exprimée sous forme d'une combinaison linéaire des objectifs dans Wise *et al.* (1997) (et les UG sont agrégées en appliquant un algorithme des k-moyennes) ; une des difficultés de cette méthode vient de l'assignation des poids respectifs à chaque membre de la fonction objectif.

2.2.2 Contrainte de contiguïté explicite : algorithmes exacts

Des méthodes exactes pour ce cas peuvent être trouvées dans la littérature. Cependant elles sont très coûteuses en temps de calcul et n'ont pour le moment été appliquées que sur de petites instances (50 UG environ).

Un problème de partitionnement de graphe avec contraintes de contiguïté a été étudié par Hansen *et al.* (2003). L'écart maximal ou "split" est défini comme la plus petite différence entre une unité d'une classe et une autre unité à l'extérieur de cette classe. Ils proposent un modèle de programmation linéaire binaire avec un nombre de contraintes exponentiel en fonction du nombre de nœuds, et un algorithme exact de génération de contraintes pour résoudre des instances d'environ 600 nœuds. Cependant leur modèle a une fonction objectif linéaire, la minimisation du "split".

Une méthode basée sur la théorie des graphes est formulée dans Zoltners et Sinha (1983). Ils modélisent la zone à découper comme un graphe d'adjacence, sélectionnent des centres de territoire (un par territoire à réaliser), et calculent les plus courts chemins entre les unités et les centres. Ils formulent alors un modèle d'optimisation linéaire (avec comme variable de décision $x_{ij} = 1$ si la région i inclut l'unité j, 0 sinon). Une unité j est assignée à un centre à un niveau d'adjacence k s'il existe une unité voisine de j déjà assignée au même centre à un niveau d'adjacence $k-1$.

Dans la thèse de Duque (2004) un modèle de sélection d'arêtes est choisi. En effet, d'après la théorie des graphes Ahuja *et al.* (1993), l'agrégation de n UG en m territoires contigus peut être réalisée en sélectionnant $n-m$ arêtes du graphe connexe. Ces $n-m$ arêtes sont une condition nécessaire mais pas suffisante. La condition supplémentaire pour la faisabilité impose que chaque paire d'unités appartenant au même territoire soit connectée par une et une seule chaîne d'arêtes. Cette contrainte est appelée contrainte d'élimination de sous-tour dans la littérature (Desrochers et Laporte (1991)). L'auteur résout une relaxation de la formulation du problème obtenu sans prendre en compte de contrainte d'élimination de sous-tour. Si la solution contient des sous-tours, ils sont éliminés en ajoutant des contraintes adéquates et en résolvant à nouveau le problème. Le processus est répété jusqu'à ce qu'une solution réalisable soit trouvée. L'objectif linéaire est la minimisation de la somme des différences entre unités assignées au même territoire :

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij} T_{ij}$$

où :

- T_{ij} variable binaire égale à 1 si les UG i et j appartiennent au même territoire ($i < j$), 0 sinon ;
- D_{ij} : valeur de la différenciation choisie entre i et j ($i < j$).

2.2.3 Contrainte de contiguïté explicite : approches heuristiques

Nous présentons dans cette sous-section six méthodes heuristiques de résolution du problème qu'on peut trouver dans la littérature.

Algorithmes basés sur la théorie des graphes

Maravalle et Simeone (1995) ont développé une méthode heuristique se basant sur la théorie des graphes. Leur objectif est de trouver une partition de plus petite inertie d'un graphe G en un nombre prédéfini p de classes. Leur méthode consiste à générer un arbre couvrant T de G, puis à générer une partition initiale de T en supprimant $p-1$ arêtes. L'ensemble de $p-1$ arêtes supprimées est ensuite modifié itérativement en choisissant d'autres arêtes de T. Les mouvements doivent améliorer un critère d'agrégation. Les auteurs introduisent aussi une quatrième étape qui consiste à générer un autre arbre couvrant T' en replaçant des arêtes de T par des arêtes non sélectionnées de G. On répète la procédure jusqu'à ce qu'on ne puisse plus trouver de mouvement qui améliore l'objectif. La contiguïté spatiale est maintenue tout au long des itérations, et on obtient toujours une solution réalisable (quelles que soient les $p-1$ arêtes supprimées).

Ricca et Simeone (2008) utilisent une méthode similaire pour générer une solution initiale dans leur algorithme de recherche locale pour résoudre un problème de découpage électoral. Leur méthode est la suivante : générer un arbre couvrant du graphe total, puis choisir aléatoirement k centres, un par district, parmi $2k$ candidats (les UG les plus peuplées). Ensuite supprimer $k-1$ arêtes de telle manière que les k sous-arbres résultants contiennent exactement un centre. Finalement, en partant de T et tant qu'il y a plus d'un centre dans le sous-arbre courant, supprimer une arête au hasard le long du chemin qui les connecte dans T. Les k districts sont les ensembles de nœuds des k sous-arbres obtenus de cette manière.

Les algorithmes de partitionnement hiérarchique adaptés

Ces algorithmes sont des heuristiques basées sur des algorithmes de partitionnement hiérarchique où seules des UG contigües peuvent être fusionnées. Le principe est d'agglomérer les UG itérativement jusqu'à obtenir le nombre de territoires désiré. Ici seules les territoires contigus peuvent être fusionnés. Cette approche hiérarchique peut être utile quand on est intéressé par des solutions imbriquées à différentes échelles (le nombre de territoires). Mais si on cherche une solution pour une échelle particulière ce n'est pas la meilleure option.

La recherche dans ce domaine utilise plusieurs méthodes d'agglomération hiérarchique, comme l'algorithme de simple lien, de lien complet, de lien moyen, et la méthode de Ward. Les caractéristiques de ces méthodes ont été étudiées par Margules *et al.* (1985). Enfin, Openshaw (1973) propose une méthode pour accélérer cette procédure de régionalisation (réduction du nombre de calculs et stockage des similarités seulement pour les UG voisines).

La méthode des régions ensemencées

Cette méthode, proposée pour la première fois par Vickrey (1961) pour résoudre les problèmes de découpage électoral, consiste pour chaque région à partir d'une UG de référence ou "graine" (souvent choisie de manière aléatoire), à laquelle on agglomère des UG voisines.

Le processus d'agrégation s'arrête lorsqu'un critère d'arrêt est satisfait : toutes les UG ont été assignées, ou un minimum de population est atteint comme par exemple dans le cas du PDE. Le processus peut être répété plusieurs fois, avec différentes UG de référence pour explorer plusieurs solutions (Thoreson et Liittschwager (1967)). Les régions peuvent être générées une par une (Rossiter et Johnston (1981)), ou en parallèle (Taylor (1973); Openshaw (1977a,b)). Les performances de l'algorithme dépendent du critère d'arrêt et de la façon de choisir les unités de référence.

Le principale difficulté de cette méthode est celle des enclaves : laisser en fin de procédure un ensemble d'UG qui ne peuvent pas former un territoire contigu (et compact dans le cas du PDE). En général, les UG restantes non assignées qui ne peuvent pas être elles-même des régions sont assignées à des UG de référence (régions existantes). Gearhart et Liittschwager (1969) proposent de nouvelles règles de terminaison, de nouvelles façons d'éviter la création d'enclaves, et différentes alternatives

pour agrandir les régions. D'autre part, Mehrotra *et al.* (1998) proposent de commencer par choisir un nœud de référence et de faire grandir les districts en commençant par les nœuds les plus éloignés du nœud de référence. L'idée est que les UG restantes en fin de procédure, toutes proches d'un nœud de référence, formeront un district compact et contigu.

Cette méthode a été reprise dans un autre article plus récent de Duque et Church (2004). Ils présentent un algorithme en deux étapes. Tout d'abord l'algorithme utilise la stratégie des régions ensemencées pour générer une solution initiale, puis l'information récupérée sur la façon dont le critère d'agrégation évolue est utilisée pour changer l'ensemble initial d'unités de référence. La meilleure solution parmi celles générées dans la première étape est choisie puis améliorée via une recherche locale utilisant une méthode de recherche avec liste tabou.

Méthode heuristique itérative de modification d'une solution initiale

Cette méthode consiste à modifier itérativement une solution initiale réalisable en tenant compte d'un critère d'agrégation. Chaque itération doit respecter les contraintes de contiguïté. Nous allons, dans un premier temps, décrire les types de mouvements qui ont été proposés dans la littérature. Ensuite nous étudierons la principale difficulté de cette méthode qui est de retirer des UG d'un territoire sans le déconnecter. Et enfin nous présenterons les différentes méthodes proposées dans la littérature pour contourner cette difficulté.

Les différents types de mouvements

Les premières méthodes proposées dans la littérature, soit Nagel (1965) pour le PDE, améliorée par Openshaw (1977a), proposent deux types de mouvements : déplacer une unité à la fois d'un territoire donneur à un territoire voisin, ou alors changer des unités une pour une entre deux territoires voisins, tout en conservant la contiguïté du territoire donneur.

De nouveaux types de mouvements sont testés par la suite dans Openshaw (1978), qui, suite à une idée introduite par Sammons (1978), incorpore la possibilité d'intégrer des fusions et des divisions de territoires (le nombre de fusions doit être égal au nombre de divisions pour maintenir le bon nombre de territoires dans la solution). Horn (1995) s'inspire aussi de cette idée d'implanter différents types de mouvements d'unités et autorise des changements temporaires du nombre de territoires (la dévia-

tion maximale autorisée par rapport au nombre réalisable de territoires est progressivement réduite à zéro). Il définit trois types de mouvements : déplacer une unité d'un territoire à un autre territoire voisin, fusionner deux territoires, sélectionner une unité pour créer un nouveau territoire. Il insiste sur l'importance de considérer des mouvements impliquant plusieurs UG à la fois.

Les différentes méthodes heuristiques

Les premières méthodes proposées, soit Nagel (1965), Openshaw (1977a), ne considèrent que les mouvements qui améliorent le critère d'agrégation. Des améliorations de cette méthode de descente sont proposées dans Openshaw et Rao (1995), qui se basent le modèle d'Openshaw (1977a) et utilisent une méthode de recherche avec tabou. Une fois qu'une unité est déplacée d'un territoire à un autre le mouvement contraire est interdit pendant un nombre fixé d'itérations. Macmillan et Pierce. (1994) proposent une méthode de recuit simulé où les mouvements n'améliorant pas l'objectif sont autorisés.

Enfin, Ricca et Simeone (2008) proposent un algorithme de recherche locale pour résoudre la formulation de théorie des graphes du problème de découpage électoral, avec un objectif multi-critère (équité de la population entre territoires, compacité). Ils comparent quatre différentes méthodes de recherche locale : un algorithme classique de descente, une méthode tabou (Glover (1990)), une méthode de recuit simulé (Kirkpatrick *et al.* (1983); Cerny (1985)) et la méthode "Old Bachelor Acceptance" décrite dans Hu *et al.* (1995). Dès que l'algorithme a atteint un minimum local, on relance la procédure à partir d'une solution réalisable générée comme expliqué dans la sous-section précédente.

Le problème de la contiguïté / Définition du voisinage

L'une des difficultés majeures de ces méthodes qui modifient itérativement une solution initiale est la contrainte de contiguïté. En effet on cherche à retirer une unité géographique sans déconnecter le territoire considéré. Comment déterminer de manière efficace si le retrait d'une UG va déconnecter ou non le territoire donneur ?

Ce problème appartient à la classe plus générale du problème de connectivité du sous-graphe, ou des problèmes dynamiques de sous-graphe. Dans ce problème il s'agit de gérer dynamiquement la suppression de nœuds (et donc des arêtes adjacentes à ce nœud) dans un graphe connexe, chaque sommet étant dans l'état "allumé" ou

"éteint". Plus précisément, il s'agit de maintenir les composantes connexes du graphe lors d'une série de requêtes et d'actualisations, tout en actualisant efficacement le graphe pour que les demandes soient plus rapides. Les requêtes consistent à vérifier si deux sommets sont connectés dans le sous-graphe induit par les sommets dans l'état "allumé".

L'article le plus récent à ce sujet dans le cas d'un graphe quelconque est celui de Duan (2010). Soit un graphe symétrique $G = (V, E)$, et $n = |V|$, $m = |E|$. La structure proposée permet de déterminer la connexité entre chaque paire de sommets actifs du sous-graphe de G induit par les sommets actifs. De plus, cette structure permet des actualisations de sommets en temps $O(m^{4/5})$ et des requêtes en temps $O(m^{1/5})$. Dans un deuxième temps ils présentent une autre structure de complexité linéaire et de complexité amortie $O(m^{2/3})$ pour les actualisations de sommets et $O(m^{1/3})$ pour les requêtes. Des algorithmes pour implémenter de telles structures sont aussi présentés dans Duan (2011).

Le problème a été introduit dans le cas des graphes planaires dans l'article de Frigioni et Italiano (2000). Dans cet article, les auteurs présentent des algorithmes efficaces pour le problème du sous-graphe dynamique, et une structure dans laquelle les sommets peuvent être rendus actifs ou inactifs en une seule actualisation sans ajouter ou supprimer les arêtes une par une. La complexité amortie est de $O(\log^3(n))$ pour des séquences d'au moins $\Omega(n)$ opérations (requête, insertion, ou suppression, allumer ou éteindre un sommet). Cependant, d'après les auteurs de cet article, les facteurs constants impliqués dans leurs bornes asymptotiques sont plutôt élevés ce qui rend leurs résultats plutôt d'intérêt théorique. Ces résultats ne sont donc pas applicables en pratique.

En ce qui concerne les articles sur le "districting", plusieurs méthodes ont été proposées. Dans les premières méthodes itératives de modification d'une solution initiale (Nagel (1965); Openshaw (1977a)), la contrainte de contiguïté d'un territoire est testée en partant d'une unité et en ajoutant successivement toutes les UG voisines. Si on a ajouté toutes les UG du territoire, celui-ci est contigu.

Macmillan et Pierce. (1994) proposent une méthode alternative pour vérifier la contiguïté appelée "switching points". Cette méthode ne considère dans une région donneuse d'unité géographique que les voisins de premier ordre de l'unité en question incluant les unités appartenant aux autres régions, au lieu d'itérer sur toutes les unités. Une preuve empirique de cette méthode est présentée dans leur article.

Ricca et Simeone (2008) formulent les deux conditions suivantes lorsqu'on veut échanger une UG entre deux districts de la solution courante tout en préservant la contiguïté :

1. le nœud à déplacer doit être adjacent à un nœud du district de destination ;
2. il ne doit pas être un point d'articulation du district d'origine.

Pour tester la condition 1, les auteurs proposent de stocker et maintenir à jour la frontière de chaque territoire (c'est-à-dire l'ensemble des nœuds du territoire adjacents à un nœud d'un autre territoire) et son cocyle (c'est-à-dire l'ensemble des arêtes qui ont un sommet appartenant au territoire et l'autre sommet à l'extérieur). Le nœud à migrer appartient à la frontière du territoire.

Pour tester la condition 2, on marque tous les nœuds adjacents à celui que l'on veut faire migrer x, on lance alors un parcours en largeur en partant de l'un deux. Si tous les nœuds marqués sont atteints au cours du parcours alors x n'est pas un point d'articulation. La recherche s'arrête dès que cette condition est satisfaite. Comme G est planaire, les auteurs indiquent que la recherche s'arrête en général assez tôt, résultant en un gain de temps considérable.

Les méthodes d'énumération et de génération de colonnes

Une autre façon de procéder consiste à formuler un modèle d'optimisation qui considère tous les territoires réalisables possibles. Mehrotra *et al.* (1998) proposent un modèle mathématique similaire à celui de Garfinkel et Nemhauser (1970), modèle d'optimisation qui nécessite d'énumérer tous les territoires réalisables comme données de l'algorithme. Leur modèle, quant à lui, peut prendre en compte beaucoup plus de territoires potentiels et les territoires réalisables sont crées via une méthode de génération de colonnes. Dans une phase finale, ils modifient la solution obtenue pour assurer l'équité de population. Dans leur méthode, ils résolvent un sous-problème linéaire, et le coût d'un territoire prend en compte sa compacité seulement. Tout district généré lors du processus respecte la contrainte de contiguïté et des bornes limites sur sa population. De plus, ils génèrent une solution initiale afin d'obtenir les valeurs duales associées aux contraintes via la technique des régions ensemencées (voir la sous-section suivante).

La méthode de génération de colonnes peut être exacte ou heuristique selon la façon dont on résout le sous-problème (voir le chapitre 3). Ici leur méthode est heu-

ristique bien qu'ils utilisent une règle de branchement exacte, car ils introduisent certaines simplifications dans leur modèle.

Dans leur modèle, ils introduisent une contrainte simplifiée de contiguïté intéressante. Enfin, ils résolvent le PDE pour des instances d'environ 50 UG.

Le sous-problème consiste en fait à minimiser un problème linéaire pour chaque UG $u \in U$ (u est alors considéré comme le centre du district généré pour des valeurs duales données). Pour s'assurer que leurs territoires générés sont contigus, ils utilisent une méthode approchée qui impose que le district soit un sous-arbre d'un arbre de plus court chemin de racine u. Cela garantit la contiguïté en éliminant cependant des districts réalisables. Mais d'après Mehrotra *et al.* (1998), les districts qui ne sont pas des sous-arbres d'un arbre de plus court chemin ont peu de chances d'être compacts. La contrainte associée est : j est sélectionné seulement si au moins un des nœuds adjacents à lui et plus proche de u est déjà sélectionné.

Modèles heuristiques mixtes

Plusieurs méthodes ont été proposées pour coupler un modèle de résolution exact avec une méthode heuristique.

Duque (2004) propose de ne pas résoudre le problème au complet d'un coup mais plutôt de sélectionner dans un premier temps un sous-ensemble de m territoires, et les UG de ces territoires sont passées dans un optimiseur pour les réagréger en m nouveaux territoires. Ensuite selon l'information obtenue, l'algorithme décide quel territoire doit entrer et quel territoire doit quitter l'optimiseur. Ce processus itératif se poursuit jusqu'à ce qu'un critère de convergence soit satisfait.

Une manière différente de procéder est proposée aussi dans Duque et Church (2004) et dans Middleton (2006). Dans ces deux articles, l'idée, insipirée de la "concentration heuristique" proposée par Rosing et ReVelle (1997), est d'utiliser l'information de multiples solutions obtenues par l'heuristique de régionalisation et de réduire le nombre de contraintes et de variables dans un modèle exact.

L'algorithme a deux étapes : d'abord lancer une heuristique de régionalisation q fois. Les m meilleures solutions sont alors sélectionnées. L'information sur les UG voisines qui ne sont jamais assignées au même territoire et sur celles qui sont toujours assignées au même territoire est ensuite utilisée pour formuler une version réduite du problème, et de le résoudre via un modèle exact.

2.3 Synthèse

Nous proposons de faire une courte synthèse sur les méthodes que nous avons évoquées dans cette revue de littérature. La méthode la plus simple à première vue consiste à utiliser un algorithme de partitionnement classique puis à modifier la solution. Le problème de minimisation de la somme des carrés a été résolu de manière exacte Aloise (2009). Cependant la contrainte de contiguïté étant très forte, la solution du problème sans celle-ci ne constituerait qu'une borne inférieure a priori de mauvaise qualité pour le PPTHC. Les méthodes de partitionnement classiques nécessitent donc d'être modifiées pour tenir compte de la contrainte de contiguïté.

Le problème de regionalisation a été résolu de manière exacte via des formulations de théorie des graphes mais seulement pour de petites instances (50 UG) et avec un objectif linéaire. De plus, la variante la plus étudiée du problème de régionalisation est celle du découpage électoral où des contraintes de compacité et sur l'équité de population entre les territoires viennent s'ajouter. Cela éloigne donc cette variante du problème qui nous intéresse, notamment en ce qui concerne les objectifs multi-critères souvent adoptés dans la littérature. Cependant si la fonction objectif peut différer de la nôtre, la méthodologie adoptée peut souvent être adaptée au problème auquel nous nous intéressons.

La minimisation de l'inertie a été traitée par des méthodes de théories de graphes Maravalle et Simeone (1995), de modification itérative d'une solution intiale Ricca et Simeone (2008), de partitionnement hiérarchique modifié, et de régions ensemencées.

A ce jour, aucune méthode exacte n'a été développée pour résoudre la minimisation d'un objectif quadratique pour un problème de régionalisation. De plus les méthodes d'énumération sont rares dans la littérature et la méthode de génération de colonnes de Mehrotra *et al.* (1998) a un objectif linéaire et ne résout pas le sous-problème à l'optimalité. De plus les instances considérées sont petites (50 UG).

Enfin, quelle que soit la méthode choisie, afin d'être implantée efficacement, la contrainte de contiguïté doit pouvoir être vérifiée rapidement. C'est pourquoi nous avons porté une certaine attention à ce sujet.

2.4 Sélection d'une méthode de résolution

La taille des instances du PPTHC à résoudre, plus 500 UG et 50 territoires, nous a imposé de développer une méthode heuristique efficace sachant que, pour une même carte, de nombreux scénarios pourraient être testés avant de sélectionner la solution à implanter. Un scénario est caractérisé par la valeur des UG et leur poids. Tester plusieurs scénarios revient à évaluer la robustesse d'une solution selon les valeurs et les poids des UG. Il convient de noter que les méthodes exactes existantes ont été appliquées sur des instances beaucoup plus petites comme on a pu le voir dans la revue de littérature.

Plusieurs choix de méthodes sont possibles, la liste étant non exhaustive :

1. utiliser des méthodes de partitionnement classiques sans prendre en compte la contrainte de contiguïté et modifier les territoires a posteriori,
2. introduire des contraintes sur les formes des territoires (ex : compacité) dans la fonction objectif et résoudre (contraintes indirectes qui donnent à la fin de la contiguïté),
3. partir d'une solution existante et la modifier en préservant la faisabilité : par exemple, faire des échanges d'unités géographiques entre territoires voisins,
4. utiliser une formulation sous forme de théorie des graphes du problème,
5. générer un grand nombre de territoires réalisables (contiguïté et poids minimal), et tenter de sélectionner un ensemble réalisable de N territoires parmi ce grand ensemble de territoires.

L'absence de contraintes sur la forme des territoires, en particulier de contraintes de compacité (comme pour le découpage électoral), pousse à ne pas considérer la méthode 2. D'autre part, la difficulté de la méthode 3 repose sur la préservation de la contiguïté lorsqu'on enlève une unité géographique d'un territoire.

Nous avons tout d'abord vu le PPTHC comme un problème de couverture d'ensemble. De plus, nous avions pour but initial de traiter des instances de l'ordre de 50.000 UG. Nous avons alors choisi de nous inspirer d'une méthodologie permettant de résoudre un problème de tournées de véhicules développée par Prescott-Gagnon *et al.* (2009). Ceux-ci utilisent une méthode heuristique de recherche à voisinage large (LNS) dans laquelle les opérateurs qui définissent le voisinage sont utilisés pour détruire la solution courante. Une heuristique dite de Branch-and-Price, reposant sur

une méthode de génération de colonnes (avec résolution du sous-problème par recherche avec tabous), est ensuite invoquée pour construire la nouvelle solution. Cette technique a déjà fait ses preuves en permettant de résoudre des problèmes complexes mais de taille relativement modeste.

Formuler le PPTHC avec un modèle de génération de colonnes revient à formuler le PPTHC en considérant implicitement tous les territoires possibles (méthode 5). La difficulté de résolution via cette formulation pour le PPTHC repose essentiellement sur la résolution du sous-problème qui permet de générer de nouveaux territoires au besoin. Toutefois, comme la méthode heuristique de génération de colonnes proposée par Prescott-Gagnon *et al.* (2009), le sous-problème peut être résolu par une heuristique quelconque qui peut s'inspirer de méthodes de recherche locale (insertion ou échange d'UG) semblables à la méthode 3. Nos résultats concluants nous ont finalement poussés à adopter ce type de méthode pour le produit final.

Chapitre 3

APPROCHE DE RÉSOLUTION

3.1 Formulation mathématique du PPTHC

La méthode proposée repose sur la formulation suivante.

- U : l'ensemble des UG,
- T : l'ensemble des territoires réalisables,
- $T' \subset T$: un sous-ensemble de territoires réalisables considérés dans le modèle,
- N : le nombre de territoires dans la solution,
- S : l'ensemble des sous-régions,
- N^s : le nombre maximal de territoires couvrant la sous-région $s \in S$,
- a_u^t : paramètre binaire égal à 1 si l'UG u est dans le territoire t, 0 sinon,
- b_{st} : paramètre binaire égal à un si le territoire t couvre la sous-région s, 0 sinon,
- Y_t : variable binaire égale à 1 si le territoire t est sélectionné dans la solution, 0 sinon,
- Ω_t : le poids du territoire t,
- Var_t : la variance de la valeur des UG du territoire t.

En utilisant cette notation, le PPTHC se formule comme suit :

$$\text{Minimiser} \quad \frac{1}{\sum_{u \in U} \omega_u} \sum_{t \in T} \Omega_t Var_t Y_t \tag{3.1}$$

$$\text{sujet à :} \quad \sum_{t \in T} a_u^t Y_t = 1, \quad \forall u \in U \tag{3.2}$$

$$\sum_{t \in T} Y_t = N \tag{3.3}$$

$$\sum_{t \in T} b_{st} Y_t \leq N^s, \quad \forall s \in S \tag{3.4}$$

$$Y_t \in \{0, 1\}, \quad \forall t \in T \tag{3.5}$$

L'objectif donné (3.1) correspond à minimiser la variance intra-territoire. Les contraintes (3.2) sont les contraintes de couverture des UG, chaque UG doit être couverte par exactement un territoire de la solution. La contrainte (3.3) est la contrainte sur le nombre de territoires dans la solution. Les contraintes (3.4) sont les contraintes de couverture des sous-zones.

Sans perte de généralité, on peut remplacer la contrainte (3.3) par

$$\sum_{t \in T} Y_t \leq N$$

puisque l'utilisation de territoires supplémentaires ne peut que faire diminuer la variance intra-territoire.

3.2 Algorithme de résolution principal

L'algorithme de résolution proposée dans ce mémoire pour le PPTHC consiste à résoudre le modèle (3.1)-(3.5) de manière heuristique, en considérant un sous-ensemble de territoires réalisables. Un premier ensemble de territoires réalisables est généré à partir d'une solution réalisable donnée en paramètre de l'algorithme. Cette solution initiale est la partition utilisée par la compagnie pour lequel ce projet a été réalisé. Deux procédures ont été mises au point pour générer des territoires réalisables :

- la première : *Guillotine*, consiste à générer de nouveaux territoires en divisant de plusieurs manières les territoires de la solution initiale et en les fusionnant.
- la seconde est une méthode de génération de colonnes qui génère itérativement de nouveaux territoires. Le générateur de colonnes est une heuristique gloutonne qui consiste à agglomérer des UG à un territoire déjà généré ou à une UG.

La méthode *Guillotine* est utilisée afin de créer un ensemble initial de territoires pour le noeud racine de la génération de colonnes. Ensuite, la méthode de génération de colonnes résout la relaxation linéaire du problème maître. Deux méthodes de branchement heuristiques détaillées plus loin ont été implémentées afin d'obtenir une solution entière.

L'algorithme 2 donne le pseudo-code de l'algorithme principal de résolution du PPTHC. On peut chercher des solutions entières à différents points de la méthode, selon les valeurs des paramètres d'entrée :

- *pOptGuillotine* : paramètre binaire égal à 1 si le problème doit être résolu en

utilisant seulement les territories initiaux et ceux générés par la Guillotine,

- $pOptCgRoot$: paramètre binaire égal à 1 si le problème doit être résolu à l'intégralité après avoir généré des colonnes dans le noeud racine de l'arbre de branchement,
- $pOptCgBB$: paramètre binaire égal à 1 si la méthode de branchement doit être appliquée afin d'obtenir une solution entière,
- $pMaxCgBB$: nombre de décisions de branchement à prendre avant de résoudre en nombres entiers.
- z_G, z_C, z_B : la valeur de l'objectif (3.1) pour les solutions trouvées respectivement via les méthodes de Guillotine, de génération de colonnes, et de branchement.

L'algorithme 2 est divisé en trois parties : la Guillotine ou initialisation (ligne 1), la génération de colonnes (lignes 3 à 9), et le branchement (lignes 11 à 20). Dans un premier temps on teste la faisabilité de la solution initiale, puis on lance la procédure ref pour générer des territoires réalisables. On résout une première fois la relaxation linéaire du problème maître et on stocke la solution obtenue. Si $pOptGuillotine = 1$, on résout le problème en nombre entiers et si le problème est réalisable, on renvoie la solution obtenue. Ensuite on lance une première fois le processus de génération de colonnes. Si $pOptCgRoot = 1$, on résout le problème en nombres entiers. Sinon on applique une décision de branchement jusqu'à ce qu'un nombre maximal de décisions de branchement $pMaxCgBB$ ait été atteint. A ce moment-là, on résout le problème en nombres entiers.

Algorithme 1: *Initialisation*

$T' = \varnothing$,
Si une solution initiale est fournie **alors**
  Vérifier la faisabilité de la solution,
  Ajouter les territoires réalisables à T', et signaler les territoires non réalisables.
  Appeler la procédure *Guillotine* pour générer de nouveaux territoires et les ajouter à T'.
  Résoudre la relaxation linéaire du modèle (3.1)-(3.5).
  Si Si une solution est trouvée **alors**
    stocker cette solution, coût z_G
  fin Si
  Si $(pOptGuillotine == 1)$ **alors**
    résoudre le problème en nombre entiers via IBM ILOG CLPEX.
    Si Si une solution est trouvée **alors**
      renvoyer la solution trouvée
    fin Si
  fin Si
fin Si

Algorithme 2: Algorithme principal

1: Générer un ensemble initial de territoires avec l'algorithme 1
2: **Si** ($pOptCgRoot == 1 || pOptCgBB == 1$) **alors**
3: Lancer la génération de colonnes pour ajouter des territoires à T' jusqu'à qu'on ne puisse plus générer de colonnes.
4: Stocker Y la solution courante du problème linéaire.
5: **Si** ($pOptCgRoot == 1$) **alors**
6: Pour l'ensemble T' courant, résoudre le modèle (3.1)-(3.5) en nombres entiers via IBM ILOG CLPEX.
7: **Si** une solution réalisable de coût z_C est trouvée et $z_C < z_G$ **alors**
8: **renvoyer** cette solution de coût z_C
9: **fin Si**
10: **Sinon Si** ($pOptCgBB == 1$ et Y n'est pas entière) **alors**
11: **Tant que** (le nombre de décisions de branchement ne dépasse pas $pMaxCgBB$) **faire**
12: Imposer une décision de branchement et retirer les colonnes incompatibles de T'.
13: Lancer la génération de colonnes pour ajouter des territoires à T'
14: **fin Tant que**
15: **Si** la solution courante est fractionnaire **alors**
16: Pour l'ensemble courant T', résoudre le modèle (3.1)-(3.5) en nombres entiers via IBM ILOG CLPEX,
17: **Si** une solution réalisable de coût z_B est trouvée et $z_B < z_G$ **alors**
18: **renvoyer** cette solution de coût z_B
19: **fin Si**
20: **fin Si**
21: **fin Si**
22: **fin Si**
23: **renvoyer** la meilleure solution réalisable et son coût

3.3 La *Guillotine*

Etant donnée une carte à partitionner et une méthode de génération de colonnes, la première étape est de générer un ensemble a priori de colonnes. Cela permet d'avoir accès d'une part aux valeurs duales des contraintes du problème et ainsi de calculer le coût réduit des colonnes générées lors de la première itération de génération de colonnes.

Nous avons choisi d'implémenter une heuristique afin d'obtenir un ensemble initial de colonnes. Cette heuristique est basée sur le principe suivant : la procédure prend comme donnée une solution initiale au problème, divise ces territoires en sous-territoires, et fusionne ces sous-territoires afin de créer de nouveaux territoires. Les sous-territoires sont seulement fusionnés par deux ou trois afin de réduire les temps de calculs (on aurait pu aussi redécouper chaque sous-territoire obtenu). Il convient de noter que l'ensemble de territoires obtenus par cette méthode peut être suffisants pour améliorer la solution initiale (voir le chapitre 5). L'algorithme 4 présente le pseudo-code de cette procédure. On note : L_1, L_2 et L_3 les listes contenant les nouveaux territoires obtenus à partir de 1, 2 ou 3 sous-territoires respectivement, et T_0 les territoires de la solution initiale.

Chaque territoire initial est divisé de plusieurs manières. Chaque découpe est déterminée par une droite du plan et la position des centroïdes (x_u, y_u) des UG du territoire à découper.

Une droite de découpe est définie par un point (x_d, y_d) et un angle θ_d. Son équation dans le plan est donc :

$$y = y_d + \tan(\theta_d)(x - x_d)$$

Comme illustré sur l'image 3.1, un territoire $t \in T_0$ est divisé selon cette droite en deux territoires t_1 et t_2 (en marron et jaune), tels que :

$$t_1 = \{u \in U \| y_u \leq y_d + \tan(\theta_d)(x_u - x_d)\}$$

et

$$t_2 = \{u \in U \| y_u > y_d + \tan(\theta_d)(x_u - x_d)\}$$

A chaque paire (t_1, t_2) on associe le score :

$$\rho_{t_1,t_2} = \omega_{t_1} Var_{t_1} + \omega_{t_2} Var_{t_2}$$

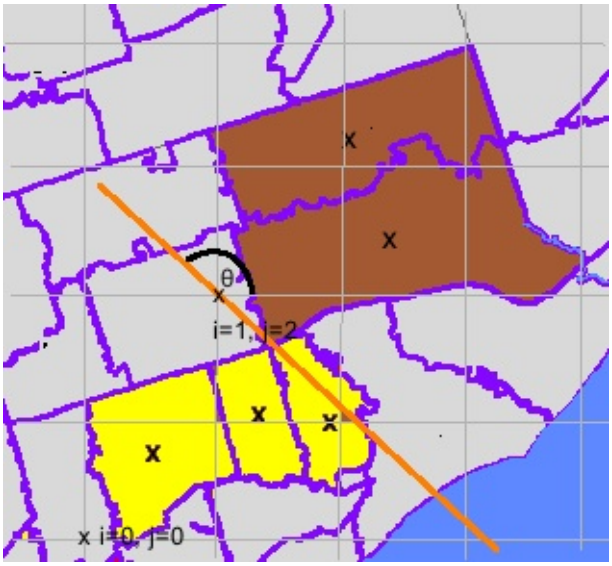

FIGURE 3.1 *Guillotine* : découpe d'un territoire

On choisit l'ensemble des droites utilisées pour générer les paires de sous-territoires à partir d'un territoire t de la manière suivante. Soit :

- $x_{min} = \underset{u \in t}{min}\ x_u$, $y_{min} = \underset{u \in t}{min}\ y_g$,
- $x_{max} = \underset{u \in t}{max}\ x_u$, $y_{max} = \underset{u \in t}{max}\ y_u$,
- $dimSpace$: nombre d'intervalles divisant les dimensions de l'espace (les axes x et y) sur le rectangle d'axes (x, y) couvrant le territoire,
- $dimAngle$: nombre d'intervalles divisant $[-\pi, \pi]$ pour les angles.

Pour définir les droites, on choisit les triplets définissant les droites $(x_d^i, y_d^j, \theta_d^k) =$

$$(x_{min} + \frac{(x_{max} - x_{min}) \times i}{dimSpace}, y_{min} + \frac{(y_{max} - y_{min}) \times j}{dimSpace}, -\frac{\pi}{2} + \frac{k}{dimAngle} \times \pi)$$

Des droites différentes peuvent conduire aux mêmes paires de sous-territoires. Dans ce cas, une seule paire est gardée. Une fois toutes les paires générées, la procédure retient les n_{paires}^{max} paires de plus bas scores ρ_{t_1,t_2} (en excluant celles qui laissent t inchangé). La division en deux sous-territoires par cette méthode peut conduire à des sous-territoires non contigus. On considère quand même ces sous-territoires, et on fera une vérification de la contiguïté en fin de procédure.

Une fois que toutes les paires de sous-territoires ont été générées et stockées dans

L_1 (lignes 1), la procédure tente de les fusionner en utilisant deux boucles. Deux territoires t_1 et t_2 sont fusionnés s'ils contiennent une UG en commun, ou s'il existe $u_1 \in t_1$ et $u_2 \in t_2$ telles que u_1 et u_2 soient adjacentes. Le nouveau territoire contient alors toutes les UG appartenant à l'un ou l'autre des territoires t_1 ou t_2. Pour fusionner deux sous-territoires, s'ils sont adjacents, on fusionne leurs deux listes de UG et on élimine les doublons. Lors de la première boucle, de nouveaux territoires sont créés en fusionnant deux sous-territoires en essayant toutes les combinaisons possibles (lignes 3 à 12). La liste de territoires L_2 est créée à l'issue de cette boucle. Lors de la seconde boucle (ligne 13 à 22), on fusionne les sous-territoires initiaux de L_1 avec ceux de L_2 et on stocke les territoires obtenus dans L_3. Enfin on vérifie la faisabilité des territoires créés et on élimine les territoires infaisables de la solution $L_1 \cup L_2 \cup L_3$.

Pour accélérer l'étape de fusion des territoires, on peut garder en mémoire pour chaque sous-territoire issu de la division la liste des sous-territoires qui lui sont voisins. Cela est réalisé lors de la première étape de fusion. Par la suite, deux territoires t_1 et t_2 sont alors fusionnés s'il existe un sous-territoire st_1 de t_1 et st_2 de t_2 tels que st_1 et st_2 soient adjacents.

Algorithme 3: Découpe de solution initiale

$L_1 = L_2 = L_3 := \varnothing$
Pour tout $t \in T_0$ **faire**
  $P := \varnothing$
  Pour $i = 0$ à $dimSpace$ **faire**
    Pour $j = 0$ à $dimSpace$ **faire**
      Pour $k = 0$ à $dimAngle - 1$ **faire**
        Créer une paire (t_1, t_2) de sous-territoires obtenus par la droite définie par le triplet $(x_d^i, y_d^j, \theta_d^k)$
        Si $(t_1, t_2) \notin P$ **alors**
          $P := P \cup \{(t_1, t_2)\}$
        fin Si
      fin Pour tout
    fin Pour tout
  fin Pour tout
  Trier les paires (t_1, t_2) dans P par ordre croissant de score ρ_{t_1,t_2}
  Garder les n_{paires}^{max} premières paires et supprimer les autres de P.
  Pour tout $(t_1, t_2) \in P$ **faire**
    $L_1 := L_1 \cup \{t_1, t_2\}$
  fin Pour tout
fin Pour tout

Algorithme 4: *Guillotine*

$L_1 = L_2 = L_3 := \varnothing$
Découper les territoires intiaux avec l'algorithme 3
Pour tout $t_1 \in L_1$ **faire**
  Pour tout $t_2 \in L_1$ tel que $t_1 \neq t_2$ **faire**
    Si ($t_1 \cap t_2 \neq \varnothing$ OU t_1 et t_2 sont voisins) **alors**
      Créer un nouveau territoire $t = t_1 \cup t_2$
      Si $t \notin L_2$ **alors**
        $L_2 := L_2 \cup \{t\}$
      fin Si
    fin Si
  fin Pour tout
fin Pour tout
Pour tout $t_1 \in L_1$ **faire**
  Pour tout $t_2 \in L_2$ **faire**
    Si ($t_1 \cap t_2 \neq \varnothing$ OU t_1 et t_2 sont voisins) **alors**
      Créer un nouveau territoire $t = t_1 \cup t_2$
      Si $t \notin L_3$ **alors**
        $L_3 := L_3 \cup \{t\}$
      fin Si
    fin Si
  fin Pour tout
fin Pour tout
Vérifier la faisabilité (contiguïté et poids minimal) des territoires de $L_1 \cup L_2 \cup L_3$ et retirer les territoires irréalisables de $L_1 \cup L_2 \cup L_3$
renvoyer $L_1 \cup L_2 \cup L_3$

3.4 Génération de colonnes

La génération de colonnes, dont l'idée générale a été présentée dans Dantzig et Wolfe (1960), est une méthode couramment utilisée pour résoudre des problèmes linéaires présentant un nombre très élevé de variables.

La formulation de cette classe de problème, dans le cas particulier d'un problème borné sans rayons extrêmes, repose sur le principe de décomposition de Dantzig-Wolfe :

$$\min_{\lambda} \sum_{x \in \Delta} c^T x \lambda_x \tag{3.6}$$

$$\sum_{x \in \Delta} Ax\lambda_x = b \tag{3.7}$$

$$\sum_{x \in \Delta} \lambda_x = 1 \tag{3.8}$$

$$\lambda_x \geq 0, \qquad \forall x \in \Delta \tag{3.9}$$

où

- Δ est l'ensemble des points extrêmes d'un polytope inclus dans $\mathbb{R}^n$
- $\lambda \in \mathbb{R}^{|\Delta|}$
- $A \in \mathbb{R}^m \times \mathbb{R}^n$
- $b \in \mathbb{R}^m$
- $c \in \mathbb{R}^n$

Dans un cadre plus général, cette formulation est modifiée pour prendre en compte les rayons extrêmes du polytope Δ. Dans ce qui suit, nous nous placerons dans le cadre du PPTHC, et on considérera que Δ est un polytope borné et donc qu'il ne contient pas de rayons extrêmes. Notons qu'un point extrême $x \in \Delta$ est associé à une variable λ_x et une colonne Ax. On utilisera indifféremment les termes "points extrême", "variable" ou "colonne" pour se référer au même objet. De plus, le modèle (3.6)-(3.9) est appelé le problème maître.

Ce type de problème est en général $NP-$difficile à cause de la contrainte d'intégrité. On résout alors pour notre application des relaxations linéaires du problème qui servent de bornes inférieures dans une méthode d'énumération implicite (branchement et évaluation).

La méthode consiste à résoudre le problème maître restreint à un sous-ensemble de colonnes $\Omega \subset \Delta$, qui est alors appelé le problème maître restreint (PMR_Ω). La solution optimale peut utiliser des points extrêmes qui ne sont pas dans Ω. On définit alors un sous-problème (SP) qui sert à générer les nouveaux points extrêmes associés à des colonnes de coût négatif à rajouter à Ω. Le problème maître restreint et le sous-problème peuvent se formuler comme suit :

(PMR_Ω)

$$\begin{aligned} \min_{\lambda} \sum_{x \in \Omega} c^T x \lambda_x \\ \sum_{x \in \Omega} Ax\lambda_x = b \\ \sum_{x \in \Delta} \lambda_x = 1 \\ \lambda_x \geq 0, \qquad \forall x \in \Omega \end{aligned}$$

(SP)

$$\overline{c} = \min_{x \in \Delta} c^T x - \alpha^T Ax - \sigma$$

où $(\alpha, \sigma) \in \mathbb{R}^m \times \mathbb{R}$ est le vecteur de variables duales associé aux contraintes de PMR_Ω.

La génération de colonnes est un processus itératif, tel que décrit dans l'algorithme 5.

Algorithme 5: Méthode de génération de colonnes.

1: Déterminer un sous-ensemble $\Omega \subset \Delta$ de variables initiales contenant une solution réalisable
2: Résoudre (PMR_Ω) par une méthode de programmation linéaire (par l'algorithme du simplexe par exemple) pour obtenir une solution primale et une solution duale.
3: Résoudre (SP).
4: **Si** $\overline{c} \geq 0$ **alors**
5: Arrêter
6: **Sinon**
7: (SP) a généré des variables de coût négatif, les rajouter à Ω
8: Retourner à l'étape 2.
9: **fin Si**

L'algorithme s'arrête alors qu'il n'a visité qu'un sous-ensemble des points ex-

trême de Δ. Une preuve d'optimalité nous est donnée par le théorème des écarts-complémentaires.

Théorème 1. *Soit λ une solution réalisable de (PMR_Δ). λ est une solution optimale de (PMR_Δ) si et seulement si il existe $\alpha \in \mathbb{R}^m$ tel que :*

$$\lambda_x(c^T x - \alpha^T A x - \sigma) = 0 \qquad \forall x \in \Delta \tag{3.10}$$

$$c^T x - \alpha^T A x - \sigma \geq 0, \qquad \forall x \in \Delta \tag{3.11}$$

Un vecteur α vérifiant les contraintes (3.10) est appelée solution duale complémentaire à la solution λ. Un vecteur α vérifiant la contrainte (3.11) est dit solution duale réalisable.

Notons Ω_f l'ensemble de points extrêmes générés lorsque l'algorithme 5 s'arrête. Comme on a résolu (PMR_f) à l'optimalité, la solution obtenue $(\lambda, \alpha, \sigma)$ vérifie les conditions (3.10) pour tout $x \in \Omega_f$. En posant $\lambda_x = 0$ pour $x \in \Delta \setminus \Omega_f$, les conditions (3.10) sont vérifiées pour tout $x \in \Delta$. De plus, le sous-problème n'a généré aucune variable de coût réduit négatif, ce qui signifie que les conditions (3.11) sont vérifiées pour tout $x \in \Delta$. On a donc obtenu une solution optimale λ à (PMR_Δ).

3.5 Formulation du sous-problème pour le PPTHC

Le sous-problème consiste à trouver une colonne de coût réduit négatif lorsqu'il en existe une ou à démontrer qu'il n'en existe pas. Pour le PPTHC, le PMR correspond à la relaxation linéaire du modèle (3.1)-(3.5) restreint à l'ensemble T des indices des variables connues. Sa résolution produit une solution duale, soit une valeur duale pour chacune de ses contraintes. Pour vérifier s'il existe une colonne de coût réduit négatif, on cherche à trouver la colonne de coût réduit minimal. Le sous-problème se formule alors comme suit.

On note par :

- α_u : valeur duale associée à la contrainte de couverture (3.2) de l'UG $u \in U$,
- μ_{nbterr} : valeur duale associée à la contrainte (3.3) sur le nombre maximal de territoires,
- α_s : valeur duale associée à la contrainte (3.4) de couverture des sous-zones.

Définissons les variables binaires suivantes :

– $x_i = 1$, si l'UG i est dans le territoire, 0 sinon.

Le coût réduit associé à une colonne $Y_{\tilde{t}}$ est le suivant :

$$c(\tilde{t}) = \Omega_{\tilde{t}} Var_{\tilde{t}} - \sum_{u \in \tilde{t}} \alpha_u - \mu_{nbterr} - \sum_{s \in S} b_{st} \alpha_s$$

En introduisant une variable μ égale à la moyenne des valeurs des UG dans le territoire, et des variables y_s permettant de déterminer si le territoire couvre la sous-zone $s \in S$, le sous-problème peut s'écrire :

$$\begin{aligned} &\min_{x,\mu,y} \textstyle\sum_{i=1}^{|U|} x_i \omega_i (v_i - \mu)^2 - \sum_{i=1}^{|U|} x_i \alpha_i - \mu_{nbterr} - \sum_{s \in S} y_s \alpha_s \\ &\Leftrightarrow \min_{x,\mu,y} \textstyle\sum_{i=1}^{|U|} x_i (\omega_i (v_i - \mu)^2 - \alpha_i) - \sum_{s \in S} y_s \alpha_s \end{aligned}$$

sujet à :

$$\sum_{i=1}^{|U|} x_i \omega_i \geq \omega_{min} \tag{3.12}$$

$$y_s \geq b_{si} x_i, \quad \forall i \in \{1, \ldots, |U|\}, \ \forall s \in S \tag{3.13}$$

$$x_i \in \{0, 1\}, \qquad \forall i \in \{1, \ldots, |U|\} \tag{3.14}$$

contraintes assurant la contiguïté du territoire. (3.15)

Afin de formuler explicitement la contrainte de contiguïté (3.15), on peut formuler notre problème comme un problème de théorie des graphes. On note alors $G(U, E)$ le graphe d'adjacence associé à la carte à découper. A chaque UG est associé un sommet appartenant à U. On numérote les UG de 1 à $|U|$, et on note E l'ensemble des arêtes entre UG voisines. Un territoire est un sous-graphe contigu de poids minimal de G.

La contrainte de contiguïté (3.15) pour un territoire t revient à vérifier qu'il existe un chemin entre toute paire de sommets de t, ce chemin étant constitué de sommets de t. Celle-ci est difficile à traiter. Pour assurer la contiguïté et formuler des contraintes adéquates, on peut construire une arborescence où un noeud artificiel o_t sert de racine pour l'arbre orienté qui contient les UG du territoire t. On remplace alors E par E^a, l'ensemble des arcs (i, j), $i \in U$, $j \in U$. Pour toute paire d'UG i et j voisines, on a deux arcs (i, j) et (j, i).

On note :

- $S \subset U$: sous-ensemble de sommets du graphe ;
- $E^a(S)$: ensemble des arcs ayant leurs deux extrémités dans un ensemble de noeuds S ;
- $\delta^-(j) = \{i|(i,j) \in E^a\}$;
- o_t : noeud artificiel utilisé dans cette formulation pour obtenir la contiguïté du territoire t ;
- $r_{ij} = 1$: si l'arc $(i,j) \in E^a$ est inclus dans l'arborescence construite pour assurer la contiguïté du territoire t, 0 sinon ;
- $TailleMax$: la taille maximale du territoire t, a priori elle est égale à $|U|-N+1$.

Les contraintes à rajouter au modèle pour assurer la contiguïté du territoire t peuvent alors s'écrire sous la forme suivante :

$$\sum_{j \in U} r_{o_t j} = 1, \tag{3.16}$$

$$\sum_{i \in \delta^-(j) \cup \{o_t\}} r_{ij} = x_j, \qquad \forall j \in U \tag{3.17}$$

$$r_{ij} + r_{ji} \leq 1, \qquad \forall (i,j) \in E \tag{3.18}$$

$$\sum_{(i,j) \in E^a(S)} r_{ij} \leq \sum_{j \in S} x_j - x_k, \qquad \forall k \in S,\ S \subseteq U,\ 2 \leq |S| \leq TailleMax \tag{3.19}$$

La contrainte (3.16) assure que le noeud racine o_t est connecté par un arc simple à son arbre spécifique de territoire. Les contraintes (3.17) requièrent que chaque UG $j \in V$ assignée au territoire t ait un degré entrant de un à l'intérieur de l'arbre de territoire spécifique. Les contraintes (3.17) et (3.18) sont des contraintes qui imposent qu'un arc soit sélectionné seulement si les deux noeuds sont dans le territoire. La contrainte (3.19) est une contrainte d'élimination de sous-tour, dans la forme présentée par Lucena et Resende (2004), qui impose que tout sous-ensemble de noeuds assignés à t induit moins d'arcs orientés dans l'arbre du territoire que le nombre de noeuds qui lui sont assignés.

3.6 Le générateur de colonnes heuristique

Dans une méthode de génération de colonnes heuristique, une heuristique peut être employée pour résoudre le sous-problème et la méthode s'arrête lorsque cette heuristique ne trouve plus de colonnes de coût réduit négatif.

Pour générer des territoires nous avons développé une heuristique décrite dans les algorithmes 6 et 7. La méthode est semblable à une méthode de régions ensemencées décrite au chapitre 2. Elle consiste à agglomérer itérativement des UG voisines à partir d'un territoire existant ou d'une UG. On note :

- $NbMaxIterUG$: taille maximale d'un territoire si on part d'une UG.
- $NbMaxIterBasis$: nombre maximal de UG à ajouter à un territoire existant.
- Var_{max} : variance maximale d'un territoire.
- $Voisinage_t$: ensemble des UG voisins du territoire courant t.
- $Voisinage_u$: ensemble des UG voisins de $u \in U$.

La procédure principale utilisée est décrite dans l'algorithme 6. On lance la méthode $GrowTerritory$ successivement sur toutes les UG de la carte ($nbBouclesUG$ fois) (lignes 2 à 14) et sur tous les territoires qui correspondent à des variables en base dans la solution de la relaxation linéaire du PMR ($nbBouclesBasis$ fois) (lignes 15 à 26). On s'arrête dans chaque cas, quand la taille du territoire créée atteint un nombre limite (lignes 5 et 17) selon que l'on soit parti d'une UG ou d'un territoire déjà existant ($NbMaxIterUG$ ou $NbMaxIterBasis$), ou lorsqu'on ne trouve plus de territoire à générer (ligne 9-10 et 21-22).

La procédure $GrowTerritory$, décrite dans l'algorithme 7, sert à ajouter une UG à un territoire existant. Pour toutes les UG u_v voisines du territoire t courant, si le coût réduit $c(t \cup \{u_v\})$ est négatif, on ajoute l'unité au territoire (ligne 8). Si on ne trouve pas d'unité à ajouter on ajoute celle qui donne un coût réduit positif minimal (ligne 18).

Afin d'introduire de la diversité dans la création des territoires, nous utilisons une méthode similaire à une méthode tabou. À chaque itération de génération de colonnes, on garde en mémoire les territoires que l'on crée au fur et à mesure dans un ensemble $Trace$ et on s'interdit d'ajouter l'unité u au territoire courant t lorsque $t \cup \{u\}$ appartient à $Trace$. On réinitialise $Trace$ à la fin de l'itération de génération de colonnes.

Enfin pour optimiser les temps de calcul, on a introduit une valeur maximale de la variance que peut atteindre un territoire, en effet si la variance est trop élevée le territoire n'a aucune chance d'améliorer notre fonction objectif. Ainsi si la variance du territoire dépasse cette valeur on arrête de lui ajouter des UG.

Cette méthode a l'avantage d'être peu coûteuse en temps de calcul et permet de s'assurer de la contiguïté d'un territoire sans avoir à faire de vérifications supplémen-

taires. Cependant sa limite est le fait qu'elle optimise de façon locale en ajoutant mais en ne retirant pas d'unité au territoire courant, c'est pour cela que l'on a introduit $Trace$ afin d'introduire plus de diversité dans notre recherche de nouvelles colonnes.

Algorithme 6: Générateur de colonnes

```
1: Trace = ∅
2: Pour (i = 1 → nbBouclesUG) faire
3:    Pour tout u ∈ U faire
4:       t = {u}
5:       Pour j = 1 → NbMaxIterUG faire
6:          t = GrowTerritory(Trace, t)
7:          Si t ≠ NULL alors
8:             Trace = Trace ∪ {t}
9:          Sinon
10:            break
11:         fin Si
12:      fin Pour tout
13:   fin Pour tout
14: fin Pour tout
15: Pour (i = 1 → nbBouclesBasis) faire
16:   Pour tout t ∈ B faire
17:      Pour j = 1 → NbMaxIterBasis faire
18:         t = GrowTerritory(Trace, t)
19:         Si t ≠ NULL alors
20:            Trace = Trace ∪ {t}
21:         Sinon
22:            break
23:         fin Si
24:      fin Pour tout
25:   fin Pour tout
26: fin Pour tout
```

Algorithme 7: $GrowTerritory(Trace, t)$

1: $c_{min} = 0, u_{min} = NULL$
2: **Pour tout** $u \in Voisinage_t$ **faire**
3: **Si** $(t \cup \{u\}) \notin Trace$ **alors**
4: $tmp = t \cup \{u\}$
5: Calculer $c(tmp)$
6: **Si** $c(tmp) < 0$ **alors**
7: **Si** tmp est réalisable **alors**
8: $T' = T' \cup tmp$
9: **fin Si**
10: Actualiser $Voisinage_{tmp}$
11: **renvoyer** tmp
12: **Sinon Si** $c_{tmp} \leq c_{min}$ **alors**
13: $c_{min} = c_{tmp}$, $u_{min} = u$
14: **fin Si**
15: **fin Si**
16: **fin Pour tout**
17: **Si** $u_{min} \neq NULL$ **alors**
18: $tmp = t \cup \{u_{min}\}$
19: Actualiser $Voisinage_{tmp}$
20: **renvoyer** tmp
21: **Sinon**
22: **renvoyer** $NULL$
23: **fin Si**

3.7 Les décisions de branchement

Nous avons implémenté deux types de décisions possibles de branchement.

Type 1 : Fixer la valeur d'une variable fractionnaire à 1 (branchement sur les territoires)

On choisit la variable Y_t de valeur fractionnaire maximale inférieure à 1 dans la solution, et on fixe de façon permanente la valeur de cette variable à 1.

On peut noter que si on fixe un nombre trop élevé de territoires dans la solution, on peut se retrouver dans une situation où le problème devient irréalisable. C'est-à-dire qu'on ne peut plus trouver avec la décision de branchement un ensemble faisable de N territoires.

Type 2 : Choisir un couple de UG, et imposer que les deux UG de ce couple appartiennent au même territoire (branchement sur les couples)

A chaque résolution du problème maître, on met à jour un tableau à double entrée *coupleUG*. Chaque case $[u_1][u_2]$ du tableau ($u_1 \in U$ et $u_2 \in U$) contient la somme des valeurs des variables associées à un territoire en solution contenant à la fois u_1 et u_2. On choisit le couple (u_1^{max}, u_2^{max}) de plus grande valeur inférieure à 1. Et on retire de l'ensemble de colonnes courant T' les territoires contenant seulement une des deux UG u_1^{max} et u_2^{max}. On impose par la suite dans la procédure $GrowTerritory$ de toujours considérer et potentiellement rajouter en même temps les deux UG du couple. On peut remarquer que si des décisions de type 2 sont appliquées plusieurs fois de suite, on peut rajouter plus de deux UG en même temps lors d'une étape de la procédure $GrowTerritory$ (algorithme 7). La probabilité de rendre le problème irréalisable est beaucoup plus faible que pour des décisions de type 1.

3.8 Résumé de l'algorithme

L'algorithme développé est un algorithme heuristique de génération de colonnes. Un ensemble initial de territoires réalisables est généré via la procédure *Guillotine* à partir d'une solution initiale T_0 donnée. La procédure *Guillotine* consiste à découper chaque territoire de T_0 de plusieurs manières puis à réaliser des fusions des sous-territoires obtenus. Des territoires réalisables sont ensuite rajoutés itérativement au

modèle en agglomérant des UG à des territoires déjà générés ou à des UG de la carte à découper. Afin de faciliter l'obtention d'une solution entière, deux décisions possibles de branchement heuristique (sans retour en arrière) ont été développées. Le problème maître est ensuite résolu en nombres entiers via IBM ILOG CLPEX.

Chapitre 4

DÉTAILS SUR L'IMPLANTATION

Nous présentons dans ce chapitre des détails de notre implantation et certaines méthodes employées pour la rendre efficace.

4.1 Informations sur les territoires

Pour chaque territoire, on maintient à jour sa liste d'unités géographiques et ses différentes caractéristiques (variance Var_t, poids ω_t, nombre de clients p_t, moyenne μ_t, sous-régions auxquelles il appartient) mais aussi un vecteur de booléens de taille égale au nombre total d'UG, qui permet de déterminer en temps constant si une UG appartient ou non à ce territoire.

4.2 La *Guillotine*

Lors de notre procédure *Guillotine*, il est nécessaire de déterminer la contiguïté d'un territoire et aussi de déterminer si deux territoires sont voisins.

4.2.1 Contiguïté

Il est nécessaire d'éliminer les territoires non contigus de la solution. Pour déterminer si un territoire est contigu on peut utiliser par exemple un parcours en largeur, ou un parcours en profondeur décrit dans l'algorithme 9 en partant d'un noeud au hasard du territoire. Le territoire est contigu si le nombre de sommets marqués après le parcours en profondeur est égal au nombre de sommets total du territoire. (On peut remarquer qu'on a construit un arbre de recouvrement du territoire en question.)

Pour chaque UG u on a une étiquette $VRAI$ ou $FAUX$ selon qu'elle a été visitée par le parcours. On commence le parcours sur une UG u_{front}. La procédure est décrite dans l'algorithme 8.

Algorithme 8: $estContigu(Territoire\ t)$

Pour tout UG $u \in t$ **faire**
  marquer u à $FAUX$;
fin Pour tout
$ParcoursEnProfondeur(t, u_{front})$
Si (tous les sommets de t ont été visités) **alors**
  renvoyer VRAI
Sinon
  renvoyer FAUX
fin Si

Algorithme 9: $ParcoursEnProfondeur(Territoire\ t, u_{current})$

Marquer $u_{current}$ à $VRAI$;
Pour tout UG $u_v \in Voisinage(u_{current}) \cap t$ **faire**
  Si g_v est marquée $FAUX$ **alors**
    $ParcoursEnProfondeur(t, u_v)$;
  fin Si
fin Pour tout

4.2.2 Territoires voisins

Pour vérifier si deux territoires sont adjacents on utilise la procédure suivante décrite dans l'algorithme 10. Dans cet algorithme, $MatriceAdj$ dénote la matrice des adjacences entre UG. On parcourt selon deux boucles imbriquées, toutes les UG de t_1 et toutes celles de t_2 jusqu'à ce qu'on trouve deux UG identiques ou voisines, les territoires sont alors voisins.

Algorithme 10: $sontVoisins(t_1, t_2)$

Pour tout $u_1 \in t_1$ **faire**
  Pour tout $u_2 \in t_2$ **faire**
    Si ($u_1 == u_2$ OU $MatriceAdj[u_1][u_2] == 1$) **alors**
      renvoyer $VRAI$
    fin Si
  fin Pour tout
fin Pour tout
renvoyer $FAUX$

4.3 Calcul des variances

Dans l'heuristique de génération de colonnes, décrite dans l'algorithme 6, on doit effectuer de nombreux calculs de variance pour calculer les coûts réduits des colonnes associées aux territoires générés ; un territoire étant généré en ajoutant une UG à un territoire existant. Ces nombreux calculs de variance peuvent s'avérer coûteux pour de grandes instances.

On rappelle que la moyenne et la variance d'un territoire sont :

$$\mu_t = \frac{\sum_{u \in t} \omega_u v_u}{\Omega_t}$$
$$\text{et} \quad Var_t = \frac{1}{\Omega_t} \sum_{u \in t} \omega_u (v_u - \mu_t)^2$$

La nouvelle moyenne et la nouvelle variance si on a ajouté $\tilde{u} \in U$ peuvent s'écrire :

$$\mu_{t \cup \tilde{u}} = \frac{1}{\Omega_t + \omega_{\tilde{u}}} (\Omega_t \mu_t + v_{\tilde{u}})$$
$$\text{et} \quad Var_t = \frac{1}{\Omega_t + \omega_{\tilde{u}}} \sum_{u \in t \cup \tilde{u}} \omega_u (v_u - \mu_{t \cup \tilde{u}})^2$$

Pour accélérer les calculs, on peut utiliser une formule de la variance, démontrée par exemple dans Apostol et Mnatsakanian (2003).

Propriété 1. *Fusion de deux territoires*

Soient deux territoires t_1 et t_2, disjoints, on a :

$$Var_{t_1 \cup t_2} = \frac{1}{(\Omega_{t_1} + \Omega_{t_2})} [\Omega_{t_1} Var_{t_1} + \Omega_{t_2} Var_{t_2} + \frac{\Omega_{t_1} \Omega_{t_2}}{\Omega_{t_1} + \Omega_{t_2}} |\mu_{t_1} - \mu_{t_2}|^2] \tag{4.1}$$

$$et \quad \mu_{t_1 \cup t_2} = \frac{1}{(\Omega_{t_1} + \Omega_{t_2})} [\Omega_{t_1} \mu_{t_1} + \Omega_{t_2} \mu_{t_2}] \tag{4.2}$$

Cette méthode est adaptée à notre méthode de résolution étant donné qu'on tient à jour la moyenne d'un territoire à chaque ajout d'une UG dans l'algorithme 7. La propriété ci-dessus, dans le cas d'une seule UG $\tilde{u} \in U$, devient :

$$Var_{t_1 \cup \tilde{u}} = \frac{1}{(\Omega_{t_1} + \omega_{\tilde{u}})} [\Omega_{t_1} Var_{t_1} + \frac{\Omega_{t_1} \omega_{\tilde{u}}}{\Omega_{t_1} + \omega_{\tilde{u}}} |\mu_{t_1} - v_{\tilde{u}}|^2]$$
$$\mu_{t_1 \cup \tilde{u}} = \frac{1}{(\Omega_{t_1} + \omega_{\tilde{u}})} [\Omega_{t_1} \mu_{t_1} + \omega_{\tilde{u}} v_{\tilde{u}}]$$

4.4 Territoires identiques

Pour notre heuristique de génération de colonnes ainsi que dans la procédure *Guillotine*, nous avons besoin d'une manière de vérifier très rapidement si le territoire que l'on s'apprête à créer n'a pas déjà été généré.

On a choisi de différencier les territoires par leur poids ω_t (ceux-ci étant des nombres moyens donc décimaux), stockés au fur et à mesure dans une structure de dictionnaire (*std* :: *map* ou *set* en C++ par exemple). Les temps de recherche et insertion étant faibles en pratique, ceci rend cette approche efficace. Par contre, il est possible en théorie de considérer deux territoires comme identiques alors qu'ils ne le sont pas.

Chapitre 5

RÉSULTATS

Ce chapitre présente les tests numériques et les résultats obtenus sur des jeux de données réelles. L'algorithme proposé au chapitre 3 a été implémenté en C++. Tous les tests ont été effectués sur un ordinateur personnel sous Linux et équipé d'un processeur AMD Opteron (Processeur 250, cpu : 2390 MHz, taille du cache : 1024 KB).

Une manière de présenter la qualité des solutions est d'utiliser le coefficient r_{intra}, défini comme la proportion de la variance totale due à la variance intra-territoire. Si on note Var_{tot} la variance totale de la valeur des UG de la carte à découper, on a :

$$r_{intra} = \frac{Var_{intra}}{Var_{tot}}$$

Ce coefficient est appelé couramment coefficient de corrélation intraclasse. Il nous permet ici de comparer plus précisément la solution initiale utilisée et la solution trouvée. Plus il est bas, meilleure est la solution.

5.1 Les données

Pour tester notre algorithme nous avons travaillé sur une carte de 522 UG, et sur sa sous-zone de 101 UG. Cela fait donc une carte de grande taille et une carte de taille réduite. Nous avons testé trois jeux de données pour la carte de 101 UG (carte 1), notés 1-1, 1-2, 1-3, et deux jeux de données pour la carte de 522 UG (carte 2), notés 2-1 et 2-2. Les données pour une même carte varient selon la valeur et le poids associés aux UG. Pour évaluer nos résultats, nous nous sommes comparés avec les territoires actuellement mis en place par la compagnie pour laquelle nous avons réalisé ce projet.

Les contraintes sont les suivantes :

– nombre maximum de territoires : $N = 55$,

– nombre maximum de territoires dans la sous-zone : $N^s = 10$.

Notre méthode permet de prendre en compte une solution initiale non réalisable. Les jeux de données 1-1, 1-2, et 1-3 sont réalisables alors que les jeux de données 2-1 et 2-2 ne sont pas tout à fait réalisables : 34 des 55 territoires dans la solution initiale sont connexes et 53 territoires sur 55 respectent la contrainte de poids minimal dans un territoire. Les jeux de données 2-1 et 2-2 sont représentatifs de solutions qui peuvent être fournies par une compagnie. En effet, il arrive que dans le cadre du PPTHC, la solution utilisée avant optimisation par une compagnie ne respecte pas toutes les contraintes. On rencontre ce cas de figure lorsque les données changent ou sont recalculées, ou lorsque le découpage en UG de la carte est modifié (nombre et/ou frontières des UG).

Tableau 5.1 Les données

Données	1-1	1-2	1-3	2-1	2-2
$\|U\|$	101			522	
N	10			55	
Taille de la sous-zone s	-			101	
N^s	-			10	

5.2 Analyse de sensibilité

Afin de déterminer les meilleurs paramètres pour l'algorithme proposé au chapitre 3, nous avons réalisé une analyse de sensibilité.

Les tableaux 5.2, 5.3, 5.4, 5.5, 5.6 et 5.7 résument les résultats obtenus sur les différentes instances en faisant varier certains paramètres de l'heuristique.

5.2.1 La *Guillotine*

Dans un premier temps, nous avons analysé les valeurs des paramètres pour la procédure *Guillotine*. Nous avons donc déterminé les meilleurs coefficients *DimSpace*, *DimAngle* (paramètres sur la taille du quadrillage), et n_{paires}^{max} (nombre de paires de sous-territoires à selectionner). Les meilleurs résultats trouvés en utilisant la *Guillotine* seule pour la carte 1 sont présentés dans le tableau 5.8.

5.2.2 Le générateur de colonnes

Un des paramètres importants de la méthode est la taille maximale des territoires générés. Elle doit être adaptée en fonction de l'instance considérée, et elle influe sur les temps de calcul. En effet une taille trop petite donne de bons temps de calcul (moins de colonnes générées) mais de moins bons résultats a priori, et une taille plus grande augmente les temps de calcul (plus de colonnes générées) mais donne de meilleurs résultats a priori. Une telle conclusion peut aussi être faite sur les nombres de boucles effectués $nbBoucleGU$ et $nbBoucleBasis$. On peut rappeler d'ailleurs qu'un nombre minimal de boucles doit être réalisé pour pouvoir tirer profit de notre liste Tabou $Trace$ mentionnée au chapitre 4.

Pour chaque carte, nous avons donc déterminé dans un premier temps la meilleure combinaison ($nbBoucleGU$, $nbMaxIterGU$), en fixant $nbBoucleBasis = 0$ lors de l'analyse. Pour cela nous avons calculé pour chaque carte la moyenne des résultats sur les instances de cette carte. Ensuite, une fois la meilleure combinaison ($nbBoucleGU$, $nbMaxIterGU$) choisie, nous avons réalisé des tests pour déterminer le meilleur couple ($nbBoucleBasis$, $nbMaxIterBasis$). Les résultats des tests effectués sont présentés dans les tableaux 5.2, 5.3, 5.4 et 5.5. Dans ces tableaux, $nbColGen$ indique le nombre de colonnes générées, et $tempsTotal$ le temps total de calcul.

Suite à l'analyse avec $nbBoucleBasis = 0$, on choisi comme combinaison ($nbBoucleGU$, $nbMaxIterGU$) celle correspondant à la moyenne de r_{intra} la plus faible sur les jeux de données d'une même carte : (2, 40) pour la carte 1, correspondant à un r_{intra} moyen de 38.1, et (3, 30) pour la carte 2, correspondant à un r_{intra} moyen de 24.3. On a ensuite fait une analyse de sensibilité sur ($nbBoucleBasis$, $nbMaxIterBasis$) de la même manière. Les meilleurs couples trouvés sont :
(0,0) pour la carte 1 et (0,0) pour la carte 2. Pour être les plus précis possible nous aurions pu calculer les résultats pour toutes les combinaisons de paramètres possibles.

On peut noter que notre algorithme est robuste à différentes valeurs des paramètres et que les résultats ne changent pas beaucoup selon ces valeurs : de 38.1 à 40.8 pour la carte 1, et de 24.3 à 25.7 pour la carte 2. Il est difficile de déterminé la corrélation entre les résultats et la valeur des paramètres, cela est dû au caractère heuristique de l'algorithme.

TABLEAU 5.2 Analyse sensibilité $nbBoucleUG$, moyenne des résultats - carte 1

$nbBouclesUG$	$nbMaxIterGU$	Var_{Intra}	r_{intra}	$nbColGen$	$tempsTotal$
1	20	3033.9	40.8	192590	728
1	30	2785.6	38.6	32426	258
1	40	2891.11	39.5	44494	878
2	20	2928.1	40.1	23871	291
2	30	2870.4	39.1	42864	892
2	40	2787.5	**38.1**	56506	312
3	20	2829.1	39.1	26368.6	132
3	30	2817.8	38.9	42310	227
3	40	2808.6	38.8	60963	448

TABLEAU 5.3 Analyse sensibilité $nbBoucleUG$, moyenne des résultats - carte 2

$nbBouclesUG$	$nbMaxIterGU$	Var_{Intra}	r_{intra}	$nbColGen$	$tempsTotal$
1	20	906.1	25.7	85439	5608.5
1	30	861.3	24.7	122068	872
1	40	854.3	24.4	162794	1600
2	20	862.3	24.7	107666	891
2	30	853.3	24.4	159341	1669
2	40	854	24.5	188565	1448
3	20	858.9	24.6	116455	810
3	30	848	**24.3**	168526	1504
3	40	854	24.4	180439	2347

TABLEAU 5.4 Analyse sensibilité $nbBoucleBasis$, moyenne des résultats - carte 1

$nbBouclesBasis$	$nbMaxIterBasis$	Var_{Intra}	r_{intra}	$nbColGen$	$tempsTotal$
1	10	2795.3	38.8	58326	1427
2	10	2777.4	38.5	61912.7	300

TABLEAU 5.5 Analyse sensibilité $nbBoucleBasis$, moyenne des résultats - carte 2

$nbBouclesBasis$	$nbMaxIterBasis$	Var_{Intra}	r_{intra}	$nbColGen$	$tempsTotal$
1	10	850.9	24.3	172358	1530
2	10	869.2	24.8	165506	5009

5.2.3 Les méthodes de branchement

Pour les deux méthodes de branchement présentées au chapitre 3, nous avons testé l'influence du paramètre $pMaxCgBB$ (nombre de décisions de branchement que l'on prend avant de résoudre le problème maître en nombres entiers via CPlex). Nous avons testé pour chaque carte deux valeurs possibles de ce paramètre selon la taille de la carte et le nombre N de territoires dans la solution. En effet, plus $pMaxCgBB$ se rapproche de N, plus la possiblité de ne pas trouver une solution réalisable augmente. Les résultats des tests sont présentés dans les tableaux 5.6 et 5.7. Les cases marquées d'un X signifient que le PMR est devenu irréalisable à cause d'un trop grand nombre de décisions de branchement. Ce problème est similaire au problème des enclaves que nous avons abordé au chapitre 2 dans le cadre des régions ensemencées.

Les méthodes de branchement développées peuvent permettre de réduire les temps de calcul sur les instances de 522 UG, par exemple pour l'instance 2-2. Elles donnent même parfois de meilleurs résulats, par exemple pour les instances 2-1. Cela est dû au fait que de nouvelles colonnes sont générées au fur et à mesure que des décisions de branchement sont imposées, ce qui permet de générer des territoires adaptés aux décisions prises. Cenpendant, dans le tableau 5.6, on peut remaquer que pour l'instance 1-2, les temps de calculs ont augmenté après avoir imposé 5 décisions de branchement au lieu de 3. Les colonnes générées en plus n'ont pas permis de compensation sur les temps de résolution du MIP. De plus les décisions supplémentaires n'ont pas été bonnes, la solution trouvée est donc de moins bonne qualité. On peut faire des commentaires semblables sur le tableau 5.7.

Suite à notre analyse de sensibilité nous avons choisi le d'imposer 10 décisions de branchement sur les couples, cette valeur donnant les meilleurs résultats sur les instance 2-1 et 2-2 (r_{intra} de 15.7 et 33.1). On peut aussi noter qu'on peut imposer plus de décisions de branchement avec la méthode de branchement sur les couples qu'avec celle sur les territoires. En effet après 20 décisions de branchement sur les UG, notre méthode de résolution du sous-problème n'arrive plus à générer de territoires qui complétent ceux déjà générés pour former une partition réalisable.

TABLEAU 5.6 Résultats - Décision de branchement sur les territoires

Données	1-1		1-2		1-3		2-1			2-2		
Var_{intra} init	1440.5		1258.3		8243.1		525.0			2067.1		
r_{intra} (%)	48.7		43.9		66.6		22.9			50.7		
Paramètres | $pMaxCgBB$	3	5	3	5	3	5	5	10	20	5	10	20
Var_{intra} trouvée	807.3	806.3	1004.1	1046.4	6650.6	X	362.9	377.2	X	1373.4	1363	X
r_{intra} (%)	27.3	27.2	35	36.5	53.7	X	15.8	16.4	X	33.7	33.3	X
Temps MIP (s)	195	8	41	8	94	X	804	1626	X	3569	1730	X
Temps total (s)	382	267	88	124	193	X	1752	2891	X	4487	4991	X

TABLEAU 5.7 Résultats - Décision de branchement sur les couples

Données	1-1		1-2		1-3		2-1			2-2		
Var_{intra} init	1440.5		1258.3		8243.1		525.0			2067.1		
r_{intra} (%)	48.7		43.9		66.6		22.9			50.7		
Paramètres | $pMaxCgBB$	5	10	5	10	5	10	10	20	30	10	20	30
Var_{intra} trouvée	881.6	1145.7	1004.1	1004.1	6612.7	6612.7	360.7	363.7	353.1	1350.5	1350.5	1346.2
r_{intra} (%)	27.4	38.7	35	35	53.4	53.4	**15.7**	15.9	15.4	**33.1**	33.1	33
Temps MIP (s)	48	31	77	36	233	79	420	317	252	345	265	244
Temps total (s)	119	127	134	109	301	197	1551	1713	2088	1484	1701	2030

5.3 Les résultats

Une solution peut être obtenue de la manière suivante :

- en utilisant seulement l'algorithme de Guillotine ("Guill"),
- par génération de colonnes suivie de la résolution exacte du problème maître ("GenCol"),
- par génération de colonnes couplée à une des deux méthodes de branchement, et suivie de la résolution exacte du problème maître ("Branch").

Pour chaque jeu de données, le tableau 5.8 indique la variance intra-territoire et le coefficient de corrélation intra-classe de la solution initiale. De plus, pour chaque variante de l'algorithme, il rapporte les mêmes statistiques, la valeur minimale obtenue au noeud racine V_{min} (qui n'est pas nécessairement une borne inférieure), le nombre total de colonnes générées $nbColGen$, le temps requis (en secondes) pour générer les colonnes et résoudre la relaxation linéaire du problème maître, le temps requis (en secondes) pour résoudre le problème maître en nombre entiers avec CPlex, et le temps total (en secondes) pour résoudre le PPTHC. Il indique aussi la variance intra obtenue ainsi que le coefficient r_{intra} correspondant. On a laissé des cases vides lorsque la méthode employée n'améliore pas la solution initiale. Les paramètres utilisés, pour chaque carte, sont ceux déterminés suite à l'analyse de sensibilité présentée dans la section précédente.

Nous n'avons pas rencontré de problème d'infaisabilité pour les 5 instances testées. De plus, les solutions obtenues sont de bonne qualité et sont réalisables contrairement aux solutions initiales données. Par exemple pour l'instance 2-1, on passe d'un r_{intra} initial de 22.9% à un r_{intra} trouvé de 15.7% (soit 31% d'amélioration) et pour l'instance 2-2, de 50.7% à 33% (soit 35% d'amélioration). De telles améliorations représentent des gains conséquents pour la compagnie, c'est donc pour cela que nous avons gardé l'algorithme de résolution présenté au chapitre 3. De plus, les temps de calcul (au maximum 35 minutes sur les instances testées) sont raisonnables et permettent de réaliser de nombreux tests successifs.

Enfin, nous avons pu remarquer que sur certaines instances (1-1, 1-3), on pouvait utiliser la méthode *Guillotine* seule pour générer des colonnes et obtenir une amélioration de la solution initiale après résolution du MIP. Les temps de calculs sont alors très faibles étant donné le peu de colonnes générées.

Les résultats obtenus nous permettent donc de valider les objectifs de recherche énoncés au chapitre 1.

5.4 Les territoires initiaux

Une compagnie peut vouloir limiter le nombre de territoires initiaux qu'elle doit modifier. En effet, effectuer un nouveau découpage implique des coûts pour chaque territoire modifié. Une fonctionnalité intéressante de notre algorithme est la possibilité d'imposer un nombre fixé de territoires initiaux dans la solution finale. Cela peut être utile si la compagnie souhaite seulement modifier en partie un plan donné. La compagnie peut alors, grâce à cette fonctionnalité, évaluer le gain apporté pour chaque territoire de la solution initiale modifié en plus, par rapport au coût de modification de sa tarification. Les territoires initiaux sélectionnés ne sont pas déterminés à l'avance, cette méthode est donc plus avantageuse que de simplement restreindre la carte à découper.

Des tests ont été effectués sur deux instances (1-1 et 1-2), et les résultats sont présentés dans le tableau 5.9. On constate que la variance intra-territoire trouvée augmente avec le nombre de territoires de la solution initiale selectionnés. Pour l'instance 1-2 on remarque qu'on peut sélectionner 3 territoires de la solution initiale sans augmenter la variance intra-territoire trouvée, une telle solution a une valeur plus élevée pour la compagnie qu'une solution de même variance intra-territoire mais qui contiendrait ici moins de 3 territoires de la solution initiale.

TABLEAU 5.8 Les meilleurs résultats trouvés

Données	1-1		1-2		1-3		2-1		2-2	
Var_{intra} init	1440.5		1258.3		8243.1		525.0		2067.1	
r_{intra} (%)	48.7		43.9		66.6		22.9		50.7	
Méthode	Guill	GenCol	Guill	GenCol	Guill	GenCol	GenCol	Branch	GenCol	Branch
Var_{intra} trouvée	1018.5	787.8	1240.9	992.5	7646.7	6582.2	362.4	360.7	1343.7	1350.5
r_{intra} (%)	34	26.6	43.3	34.6	61.8	53.2	15.8	15.7	33	33.3
$nbColGen$	4913	54024	16149	41292	14610	74203	158967	100637	173671	106898
Temps MIP (s)	3	95	14	52	9	216	1424	420	918	345
Temps total (s)	7	457	19	111	13	468	1850	1551	1770	1484

TABLEAU 5.9 Solution trouvée selon le nombre minimum de territoires initiaux conservés carte 1

Données		1-1	1-2
r_{intra} init		48.7	43.9
Nombre	1	26.5	34.4
de territoires initiaux	2	28.2	34.4
conservés	3	33.3	34.4
.	4	33.3	35
.	5	34.0	35
.	6	36.8	35
.	7	41.9	35

Chapitre 6

CONCLUSION

L'essentiel de nos travaux réside dans :

- le développement d'une heuristique pour générer un ensemble initial de territoires
- le développement d'une heuristique rapide de génération de colonnes pour résoudre des instances du PPTHC de plus de 500 UG,
- ainsi que la mise en place de méthodes de branchement.

Les solutions trouvées sont de très bonne qualité (33% d'amélioration par rapport aux résultats de la compagnie) et les temps de calculs raisonnables (en moyenne 30 minutes pour les instances de plus de 500 UG). Ces résultats nous ont donc permis de valider nos objectifs de recherche qui étaient de développer une méthode de résolution efficace et peu coûteuse en temps de calcul afin de pouvoir réaliser de nombreux tests successifs et de servir ainsi d'outil d'aide à la décision. Nous avons aussi introduit une fonctionnalité très intéressante pour la compagnie, celle de fixer à l'avance un certain nombre de territoires initiaux dans la solution finale. Cette fonctionnalité permet de contrôler le nombre de territoires initaux à modifier (on modifie alors le plan initial seulement en partie), et donc de réduire au final les coût engendrés par la mise en place du nouveau découpage.

Une des limitations de notre méthode est que nous n'avons pas pu la tester pour le moment sur de nombreuses instances. Il est donc possible a priori de ne pas améliorer la solution initiale. Connaître la solution optimale de chaque instance afin d'évaluer la qualité de nos solutions pourrait aussi être utile, même si notre but n'est pas de résoudre le problème de manière exacte. Enfin, il serait intéressant de comparer notre méthode avec d'autres méthodes déjà développées (les méthodes utilisant un modèle de théorie des graphes par exemple).

Pour conclure ce mémoire, nous présentons plusieurs pistes qu'il pourrait être intéressant d'explorer :

- inclure lors du processus de génération de colonnes, d'autres manières de générer des territoires comme : la possibilité de retirer des UG d'un territoire en base par exemple, en s'inspirant des méthodes indiquées dans la revue de littérature,
- inclure une méthode de recherche à voisinage large, avec des mouvements comme la fusion et la découpe de territoires, les échanges d'UG entre territoires voisins, le retrait d'UG d'un territoire, pour modifier la solution obtenue par la génération de colonnes et répéter le processus,
- utiliser la méthode à plusieurs reprises pour réoptimiser des parties de la solution obtenue,
- résoudre le sous-problème de manière exacte.

Références

AHUJA, R., MAGNANTI, T. et ORLIN, J. (1993). *Network flows : Theory, algorithms, and applications*. Prentice-Hall International, Inc.

ALOISE, D. (2009). *Exact Minimum Sum of Square Clustering*. Thèse de doctorat, École Polytechnique de Montréal.

ALTMAN, M. (1997). Sums of squares of distances in m-space. *Rutgers Comput. and Technical Law J 23*, vol. 81.

APOSTOL, T. et MNATSAKANIAN, M. (2003). Sums of squares of distances in m-space. *Amer. Math. Monthly*, vol. 110, pp. 516–526.

BACAO, F., LOBO, V. et PAINHO, M. (2005). Applying genetic algorithms to zone design. *Soft Computing*, vol. 9, pp. 341–348.

BERGEY, P., RAGSDALE, C. et HOSKOTE, M. (2003). A decision support system for the electrical power districting problem. *Decision Support Systems*, vol. 36, pp. 1–17.

BLAIS, M., LAPIERRE, S. et LAPORTE, G. (2003). Solving a home-care districting problem in an urban setting. *The Journal of the Operational Research Society*, 54, pp. 1141–1147.

CERNY, V. (1985). A thermodynamical approach to the travelling salesman problem : an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, vol. 45, pp. 41–55.

D'AMICO, S., WANG, S., BATTA, R. et RUMP, C. (2002). A simulated annealing approach to police district design. *Computers & Operations Research*, vol. 29, pp. 667–684.

DANTZIG, G. et WOLFE, P. (1960). Decomposition principle for linear programs. *Decomposition principle for linear programs*, vol. 8, pp. 101–111.

DESROCHERS, M. et LAPORTE, G. (1991). Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, vol. 10, pp. 27–36.

DUAN, R. (2010). New data structures for subgraph connectivity. *ICALP Õ10 : 37th International Colloquium on Automata, Languages and Programming*, pp. 201–212.

DUAN, R. (2011). *Algorithms and Dynamic Data Structures for Basic Graph Optimization Problems*. Thèse de doctorat.

DUQUE, J. (2004). *Design of homogeneous territorial units. A methodological proposal and applications*. Thèse de doctorat, University of Barcelona.

DUQUE, J. et CHURCH, R. (2004). A new heuristic model for designing analytical regions. *North American Meeting of the International Regional Science Association, Seattle, WA*.

DUQUE, J., RAMOS, R. et SURINACH, J. (2007). Supervised regionalization methods : a survey. *International Regional Science Review*, vol. 30, pp. 195–220.

FERLAND, J. et GUÉNETTE, G. (1990). Decision support system for the school districting problem. *Operations Research*, vol. 38, pp. 15–21.

FRIGIONI, D. et ITALIANO, G. (2000). Dynamically switching vertices in planar graphs. *Algorithmica*, vol. 28, pp. 76–103.

GARFINKEL, R. et NEMHAUSER, G. (1970). Optimal political districting by implicit enumeration techniques. *Management Science Series B-Application*, vol. 16, pp. 495–508.

GASCON, V., GORVAN, T. et MICHELON, P. (2010). Districting problem for a public medical clinic . *En préparation pour soumission à INFOR*.

GEARHART, B. et LIITTSCHWAGER, J. (1969). Legislative districting by computer . *Behavioral Science*, vol. 14, pp. 404–5417.

GLOVER, F. (1990). Tabu search - Part II . *ORSA Journal on Computing 2*, pp. 4–32.

GORDON, A. (1996). A survey of constrained classification. *Computational Statistics & Data Analysis*, vol. 21, pp. 17–29.

GORDON, A. (1999). *Classification. 2nd ed.* Boca Raton, FL : Chapman & Hall-CR.

HANSEN, P., JAUMARD, B., MEYER, C., SIMEONE, B. et DORING, V. (2003). Maximum split clustering under connectivity constraints. *Journal of Classification*, vol. 20, pp. 143–180.

HESS, S., WEAVER, J., SIEGFELD, H., WHELAN, J. et ZITLAU, P. (1965). Nonpartisan political redistricting by computer. *Operations Research*, vol. 13, pp. 998–1006.

HORN, M. (1995). Solution techniques for large regional partitioning problems. *Geographical Analysis*, vol. 27, pp. 230–48.

HU, T., KAHNG, A. et TSAO, C. (1995). Old bachelor acceptance : A new class of non-monotone threshold accepting methods. *ORSA Journal on Computing*, vol. 7, pp. 417–425.

KIRKPATRICK, S., GELATT, C. J. et VECCHI, M. (1983). Optimization by simulated annealing. *Science*, vol. 220, pp. 671–680.

LUCENA, A. et RESENDE, M. (2004). Strong lower bounds for the Prize Collecting Steiner Problem in graphs. *Discret App Math*, vol. 141, pp. 277–294.

MACMILLAN, W. et PIERCE., T. (1994). Optimization modelling in a GIS framework : The problem of political redistricting. *Spatial analysis and GIS*, pp. 221–46.

MACQUEEN, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–97.

MARAVALLE, M. et SIMEONE, B. (1995). A spanning tree heuristic for regional clustering. *Communications in Statistics-Theory and Methods*, vol. 24, pp. 625–39.

MARGULES, C., FAITH, D. et BELBIN, L. (1985). An adjacency constraint in agglomerative hierarchical classifications of geographic data. *Environment and Planning*, vol. 17, pp. 397–412.

MEHROTRA, A., JOHNSON, E. et NEMHAUSER, G. L. (1998). An optimization based heuristic for political districting. *Management Science*, vol. 44, pp. 1100–14.

MERLE, O., HANSEN, P., JAUMARD, B. et MLADENOVIĆ, N. (2000). An interior point algorithm for minimum sum-of-squares clustering. *SIAM Journal on Scientific Computing*, vol. 21, pp. 1485–1505.

MIDDLETON, R. (2006). *Geographical distillation : Application of the p-median, traveling salesman, and regionalization problems*. Thèse de doctorat, University of California at Santa Barbara.

MURTAGH, F. (1985). A survey of algorithms for contiguity-constrained clustering and related problems. *Computer Journal*, vol. 28, pp. 82–88.

MUYLDERMANS, L., CATTRYSSE, D., OUDHEUSDEN, V. et LOTAN, T. (2002). Districting for salt spreading operations. *European Journal of Operational Research*, vol. 139, pp. 521–532.

NAGEL, S. (1965). Simplified bipartisan computer redistricting. *Stanford Law Review*, vol. 17, pp. 863–899.

OPENSHAW, S. (1973). A regionalisation program for large data sets. *Computer Applications*, vol. 3, pp. 136–147.

OPENSHAW, S. (1977a). A geographical solution to scale and aggregation problems in region-building, partitioning and spatial modeling. *Transactions of the Institute of British Geographers*, vol. 2, pp. 459–72.

OPENSHAW, S. (1977b). Optimal zoning systems for spatial interaction models. *Environment and Planning*, vol. 9, pp. 169–84.

OPENSHAW, S. (1978). An optimal zoning approach to the study of spatially aggregated data. *Spatial representation and spatial interaction*, pp. 95–113.

OPENSHAW, S. et RAO, L. (1995). Algorithms for reengineering 1991 census geography. *Environment and Planning*, vol. 27, pp. 425–46.

OPENSHAW, S. et WYMER, C. (1995). Classifying and regionalizing census data. *Census users handbook*, pp. 239–70.

P. HANSEN, N. M. (2001). J-means : a new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, 34, pp. 405–413.

PARK, K., LEE, K., PARK, S. et LEE., H. (2000). Telecommunication node clustering with node compatibility and network survivability requirements. *Management Science*, vol. 46, pp. 363–74.

PRESCOTT-GAGNON, E., DESAULNIERS, G. et ROUSSEAU, L. (2009). A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, vol. 54, pp. 190–204.

RICCA, F. et SIMEONE, B. (2008). Local search algorithms for political districting. *European Journal of Operations Research*, vol. 189, pp. 1409–1426.

ROSING, K. et REVELLE, C. S. (1997). Heuristic concentration : Two stage solution construction. *European Journal of Operational Research*, vol. 97, pp. 75–86.

ROSSITER, D. et JOHNSTON, R. J. (1981). rogram GROUP - The identification of all possible solutions to a constituency-delimitation problem. *Environment and Planning*, vol. 13, pp. 231–238.

RYAN, D. et FOSTER, B. (1981). An Integer Programming Approach to Scheduling. *A. Wren (ed.), Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, vol. 54, pp. 269–280.

SAMMONS, R. (1978). A simplistic approach to the redistricting problem. *Spatial representation and spatial interaction*, pp. 71–94.

TAVARES-PEREIRA, F., FIGUEIRA, J., MOUSSEAU, V. et ROY, B. (2007). Multiple criteria districting problems. *Annals of Operations Research*, 154, 69–92. 10.1007/s10479-007-0181-5.

TAYLOR, P. (1973). Some implications of spatial organization of elections. *Transactions of the Institute of British Geographers*, vol. 60, pp. 121–136.

THORESON, J. et LIITTSCHWAGER, J. (1967). Legislative districting by computer simulation. *Behavioral Science*, vol. 12, pp. 237–47.

VICKREY, W. (1961). On the prevention of gerrymandering. *Political Science Quarterly*, vol. 76, pp. 105–110.

WEAVER, J. et HESS, S. W. (1963). A procedure for nonpartisan districting : Development of computer techniques. *Yale Law Journal*, vol. 73, pp. 288–308.

WISE, S., HAINING, R. P. et MA, J. (1997). Regionalisation tools for exploratory spatial analysis of health data. *Recent developments in spatial analysis : Spatial statistics, behavioural modelling, and computational intelligence*, pp. 83–100.

ZOLTNERS, A. et SINHA, P. (1983). Sales territory alignment : A review and model. *Management Science*, vol. 29, pp. 1237–56.