Université De Versailles St-Quentin-En-Yvelines
Université Paris-Saclay

MÉMOIRE DE MASTER

Présenté en vue de l'obtention du
Diplôme de master de recherche en Informatique
Option : Algorithmique et Modélisation à l'Interface des Sciences

Par
Oumayma KAABI

Gestion de collecte des déchets post-inondation

Sous la direction de :
Mme. Sandrine VIAL
M. Serge LHOMME
M. Arnaud LE GUILCHER
M. Mickael BRASEBIN

1er Avril 2016 — 30 Septembre 2016

Dédicace

À mon père, mon âme et ma raison de vivre

À ma mère, ma chére bien aimée et joie de ma vie

À mon frère et mes sœurs, mon sourire et mon cœur

À mes amis et mes proches

Remerciements

Mes premiers remerciements s'adressent principalement à mon équipe encadrante qui m'a dirigée et m'a guidée tout au long de la réalisation de ce travail.

Je tiens tout d'abord à exprimer ma profonde gratitude et reconnaissance à Serge Lhomme pour l'énorme soutien scientifique et moral qu'il m'a accordé pendant le déroulement de ce stage.

Je tiens également à remercier chaleureusement Mickael Brasebin, et Arnaud le Guilcher pour leurs qualités humaines et scientifiques et pour leurs précieux conseils, qui ont contribué à la réalisation de ce travail.

Je remercie également tous les membres du laboratoire COGIT qui m'ont offert un environnement de travail agréable et propice aux échanges scientifiques.

Que les membres de jury reçoivent mes sincères remerciements pour l'honneur qu'ils m'ont fait en acceptant de juger mon travail.

Dans la crainte d'oublier quelqu'un, je tiens à remercier toutes les personnes qui ont contribué de prés ou de loin à la réalisation de ce travail.

AVANT-PROPOS

Dans le cadre de l'obtention de mon diplôme de Master de recherche intitulé par : Algorithmique et Modélisation à l'Interface des Sciencs **AMIS**, de l'Université de Versailles Saint-Quentin en Yvelines **UVSQ/Paris Saclay**, ce rapport de mémoire a été réalisé suite un stage de recherche qui a duré six mois et qui a été effectué au sein du laboratoire **COGIT** (Cartographie et Géomatique) de l'Institut Géographique National **IGN**, sur un projet nommé SIGOPT coordonné par le laboratoire d'urbanisme **lab'urba**.

Le **Lab'urba** est une équipe d'accueil qui a été créée en 2008, elle est composé d'un ensemble des chercheurs en sciences humaines et sociales et sciences de l'ingénieur qui travaillent sur les espaces urbains et dans le champ de l'aménagement et de l'urbanisme.

Ce laboratoire est organisé en quatre équipes thématiques :

— Politiques urbaines et développement territorial ;
— Inégalités, discriminations ;
— L'urbanisme : idées, méthodes, acteurs ;
— Génie urbain et environnement.

Créé en 1988, le laboratoire **COGIT** constitue l'un des services de recherche de l'IGN qui comporte trois autres laboratoires :

— **LOEMI** : Laboratoire d'électronique et de micro-informatique ;
— **LAREG** : Laboratoire de recherche en géodésie ;
— **MATIS** : Laboratoire de méthodes d'analyse et de traitement d'image pour la stéréo-restitution.

COGIT est composé par une équipe de 29 membres, ingénieurs et des doctorants. Cette équipe est en charge des recherches liées à la gestion, la diffusion, la représentation ainsi que l'utilisation des données topographiques vectorielles. Les travaux du laboratoire s'articulent autour de trois axes de recherche :

— L'analyse de l'espace et de ses dynamiques ;
— La qualification et interopérabilité des référentiels de données géographiques ;
— La visualisation, cartographie et interactions.

Ce stage a eu lieu dans le premier axe : "L'analyse de l'espace et de ses dynamiques", qui est animé par Mickael Brasebin. Cet axe de recherche a pour objectif la proposition des modèles spatio-temporels facilitant :

- L'exploration de données ;
- L'analyse des dynamiques du territoire ;
- La prise en compte fine du territoire dans l'analyse de phénomènes et leurs dynamiques.

Table des matières

Table des figures

Liste des tableaux

Abréviations

AP	Allocation Problem
CEPRI	Centre Européen de Prévention des Risques d'Inondation
CVRP	Capacitated Vehicule Routing Problem
LP	Location Problem
MECADEPI	Méthode d'Évaluation et de Caractérisation des Déchets Post-Inondation
MEDDE	Ministère de l'Écologie du Développement Durable et de l'Énergie
MIP	Mixed Integer Programming
PL	Programmation Linéaire
RO	Recherche Opérationnelle
SIG	Système d'Information Géographique
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem

Chapitre 1

Contexte de recherche et objectifs de stage

Une inondation est un phénomène naturel qui correspond à la submersion temporaire d'une zone qui est considérée sèche. Ce phénomène constitue l'un des risques naturels majeurs dans le monde entier, il provoque des dommages et des perturbations sur les infrastructures, des pertes économiques et parfois entraine des pertes en vies humaines.

Selon les informations fournies par le Centre Européen de Prévention des Risques d'Inondation [3], l'inondation apparaît comme la plus importante menace naturelle en France en terme de nombre d'occurence puisqu'elle représente 80% des catastrophes naturelles. Les statistiques ont montré que 15000 communes françaises et environ 17 millions de personnes sont exposées aux inondations [4].

En région parisienne, l'inondation la plus importante fut celle de 1910, la ville fut inondée pendant deux semaines et le nombre des sinistrés s'éleva à 200000 personnes. Entre les années 1955 et 2010 la hauteur de la Seine a atteint un niveau entre 5 et 7 mètres ce qui a entrainé l'inondation des quais. La dernière inondation a eu lieu en juin 2016, une montée inhabituelle des eaux, durant laquelle, au niveau du pont d'Austerlitz, la Seine avait atteint un record de hauteur de 8,62 mètres. La figure 1.1 montre la crue de la seine entre 1910 et 2016.

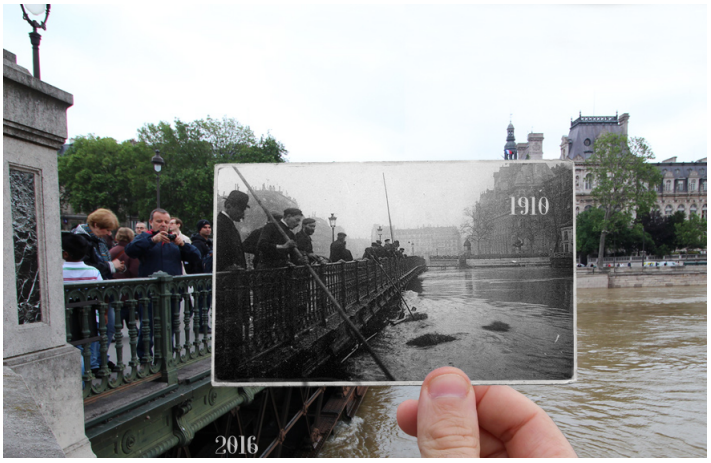

FIGURE 1.1 – Crue de la seine entre 1910 et 2016 [1]

L'importance de la force des inondations et leur impact sur les infrastructures ont été prouvés par les chiffres fournis par le CEPRI [3]. De plus que les inondations génèrent des quantités très importantes de déchets. Ainsi, la quantité de déchets produite dépasse souvent les capacités de prise en charge des territoires. À Prague, en 2012, 270000 tonnes de déchets ont été produites, leur élimination a nécessité 11 mois, alors qu'à Dresde, la quantité produite a atteint une quantité équivalente à trois années de collecte. Lors de la tempête Xynthia, en 2010, la production de déchets s'est élevée pour certaines communes à environ 12 années de production normale.

Les déchets produits par l'inondation sont de différents types. La notion de déchets post-catastrophe a été défini par Bonnemains[5] comme : « tous les matériaux, matières, objets et dépôts qui à la suite d'une catastrophe naturelle ou technologique sont impropres à la consommation, inutilisables en l'état, susceptibles d'avoir un impact sur l'environnement, la santé humaine, la salubrité publique ou de porter atteinte à la biodiversité».

Les quantités des déchets produites ont des impacts à deux niveaux :

— des impacts fonctionnels sur le territoire à court terme : elles bloquent l'accès aux installations et aux ressources critiques ;
— des impacts sanitaire et environnementaux à long : elles créent un environnement favorable au développement d'espèces nuisible et favorisent la contamination du sol, de l'eau ainsi que l'air.

Selon Robin des Bois [6], suite au passage de Katrina, à la Nouvelle-Orléans, des niveaux élevés de pollution de l'air et du sol ont été observés, le nombre de spores en milieu ouvert est passé de 12000 spores par m^3 jusqu'à 50000 spores par m^3 et parfois jusqu'à 650000 spores par m^3 dans certaines habitations.

La gestion des déchets générés par une inondation est une étape cruciale pour la reprise des activités économiques et sociales des territoires. Afin de gérer au mieux ces quantités importantes de déchets, il est intéressant de pouvoir les quantifier le plus précisément possible avant que l'inondation se produise. Cette estimation est en effet nécessaire pour pouvoir optimiser la collecte de ces déchets.

Pour ce faire, le projet MECADEPI (Méthode d'évaluation et de caractérisation des déchets post-inondation) a été réalisé. La méthode développée se base sur une connaissance des équipements des maisons, des établissements, ainsi que les entreprises qui sont considérés comme les sources de production de déchets pour évaluer ces quantités. Cette méthode permet aux gestionnaires du territoire d'acquérir de meilleures connaissances concernant les principaux types de déchets et les quantités associées [3].

Bien que cette méthode ait permis de faciliter la quantification et la caractérisation des déchets, il est nécessaire de souligner que les résultats de cette méthode reposent sur des sources de données géographiques hétérogènes et imparfaites, ce qui impacte la qualité des résultats produits.

Dans ce contexte, le projet SIGOPT (Système d'Information Géographique et Optimisation Territoriale) fiancé par la COMUE Paris-Est et le CNRS, a été réalisé. Ce projet se focalise sur la collecte des déchets post- inondation et son objectif principal est de produire une extension à un Système d'Information Géographique (SIG) permettant l'optimisation de la collecte des déchets post-inondation. Ce projet s'articule autour de trois axes : un premier axe qui sert à l'automatisation de la méthode MECADEPI, un deuxième axe qui a pour but la collecte des déchets générés et finalement un axe pour analyser et évaluer la qualité des données.

La figure 1.2 montre l'articulation entre ces trois axes.

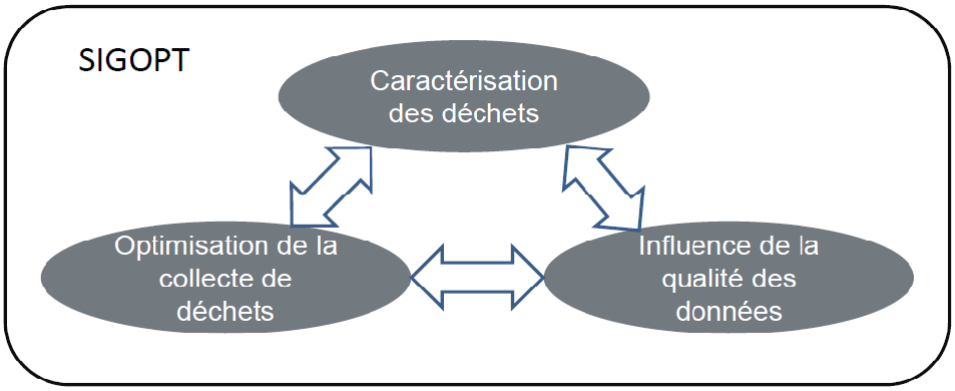

FIGURE 1.2 – Les trois axes du projet SIGOPT

Le but de notre travail présenté dans ce mémoire s'inscrit dans le deuxième axe du projet SIGOPT : proposer des méthodes d'optimisation pour la résolution de problème de gestion de collecte de déchets post-inondation tout en respectant les contraintes de capacité et de coût.

Ce mémoire est composé de trois chapitres organisés comme suit : un premier chapitre introductif qui représente le contexte et les objectifs de notre stage. Le deuxième chapitre fournit un état de l'art sur les problèmes de collecte de déchets post-inondation et spécifiquement sur les problèmes de collecte des ordures ménagères. Le troisième chapitre, présente une description de notre problème et décrit les méthodes de résolution sur lesquelles nous nous sommes fondées pour la résolution de notre problème : la première méthode est une méthode exacte basée sur la programmation linéaire mixte (MIP) et la deuxième méthode est une méthode approchée, une métaheuristique basée sur le recuit simulé. Finalement, ce mémoire se termine par une conclusion sur les travaux réalisés.

Chapitre 2

État de l'art

2.1 Introduction

Après une inondation, une réponse rapide face aux quantités importantes de déchets produites doit être faite pour maintenir le fonctionnement des services dans les territoires. Dans les faits, cette problématique est liée à deux problèmes classiques d'optimisation combinatoire : les problèmes de localisation pour déterminer l'emplacement optimal des centres de stockages et les problèmes de tournées de véhicules pour optimiser le réseau de la collecte. Dans ce chapitre nous allons présenter un état de l'art de ces deux problèmes, puis nous nous focaliserons sur le problème de collecte de déchets.

2.2 Les problèmes de logistique du transport

2.2.1 Problème de localisation et d'allocation

Le problème de localisation et d'allocation consiste à localiser un certain nombre d'entrepôts sur un ensemble de sites dans l'objectif de servir un ensemble de clients à un coût de transport minimal [7]. Dans le but de minimiser les coûts de transport, trois éléments doivent être identifiés : le nombre de sites à ouvrir, la localisation de ces sites et finalement la façon d'affecter l'ensemble des clients à ces sites.

Ce problème a été largement étudié et plusieurs formulations ont été proposées. On peut citer celle de Efroymson et Ray [8] : étant donné E l'ensemble des n sites et J l'ensemble des m clients, la formulation utilise une variable c_{ij} qui représente le coût d'affectation du client j à l'entrepôt i, une variable f_i qui représente le coût de localisation d'un entrepôt sur le site i ainsi qu'une variable binaire x_{ij} qui est égale à 1 si le client j est affecté à l'entrepôt i, 0 sinon et une variable binaire y_i qui est égale à 1 si un entrepôt est installé sur le site i et 0 si ce n'est pas le cas.

Fonction-Objectif :

$$\min \sum_{i=1}^{J} \sum_{i=1}^{E} c_{ij} x_{ij} \sum_{i=1}^{E} f_i y_i \tag{2.1}$$

Contraintes

$$\sum_{i=1}^{E} x_{ij} = 1 \qquad \forall j \in J \tag{2.2}$$

$$x_{ij} \leq y_i \qquad \forall j \in J, \forall i \in E \tag{2.3}$$

$$y_i \in \{0, 1\} \qquad \forall i \in E \tag{2.4}$$

$$x_{ij} > 0 \qquad \forall j \in J, \forall i \in E \tag{2.5}$$

La fonction objectif (2.1) vise à minimiser la somme des coûts. Les contraintes (2.2) obligent à chaque client d'être affecté qu'à un seul site et les contraintes (2.3) imposent que chaque client j ne puisse être servi par le site i que si un entrepôt y est installé. Les contraintes(2.4) mettent en place une restriction binaire sur la variable de décision y_i. Finalement, les contraintes (2.5) sont les contraintes de non-négativité sur les variables x_{ij}.

2.2.2 Problème du voyageur de commerce

Le problème de voyageur de commerce, plus connu sous son nom en anglais Traveling Salesman Problem (TSP), est un problème d'optimisation combinatoire qui appartient à la classe des problèmes NP-Complets. Ce problème consiste à déterminer un trajet minimal permettant à un voyageur de visiter n villes avant de retourner à son point de départ, tout en passant une et une seule fois par chaque ville, et en minimisant la distance totale parcourue [9]. Dans un graphe, la résolution de ce problème revient trouver un cycle hamiltonien(un cycle qui passe exactement une fois par chaque sommet d'un graphe) de coût minimal.

Le problème de voyageur de commerce peut être modélisé comme étant un graphe complet G = (V, E) où $V = \{V_1.., V_n\}$ est l'ensemble de sommets modélisant les villes et $E = \{[V_i, V_j] : i \leq j; V_i, V_j \in V\}$ est l'ensemble d'arêtes reliant les villes.

La figure 2.1 illustre le principe de problème de voyageur de commerce

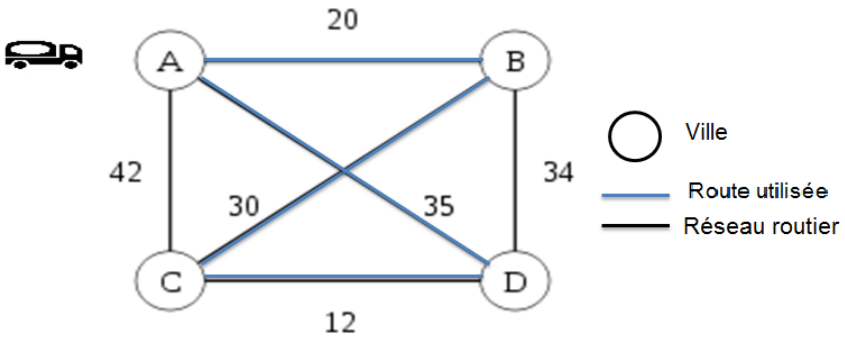

FIGURE 2.1 – Problème de voyageur de commerce

2.2.3 Problème des tournées des véhicules

Définition Le problème de tournées de véhicules nommé en anglais Vehicle Routing Problem (VRP) est un problème de logistique du transport. Étant introduit en 1959 par Dantzig et Ramser [10], ce problème appartient au problème NP-difficile puisqu'il n'existe pas d'algorithme en temps polynomial pour le résoudre. Le VRP est une extension du problème de voyageur de commerce (TSP) et le problème du bin-packing où le but est d'économiser le rangement d'un ensemble d'articles dans des boîtes [11].

L'objectif de ce problème est la satisfaction d'un ensemble de demandes de plusieurs clients dispersés dans un réseau, à partir d'un ou plusieurs dépôts, à l'aide des véhicules de capacité limitée tout en minimisant le coût total de tournées.

La figure 2.2 illustre le principe de problème de tournées de véhicules.

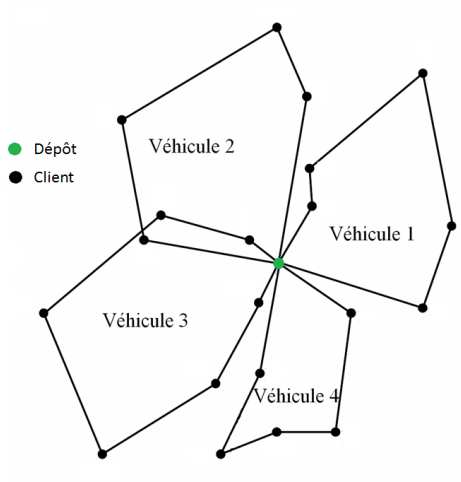

FIGURE 2.2 – Problème de tournées de véhicules

Formulation du problème Dans la littérature, plusieurs formulations du problème de VRP ont été proposées. On peut citer celle de Semet [12] : soit G = (V, E) un graphe complet où $V = \{V_0, V_1.., N\}$ est un ensemble de sommets modélisant les villes (ou les clients) et E =$\{[i, j] : i, j \in V\}$ est l'ensemble d'arêtes reliant les villes. Le dépôt est représenté par le sommet V_0, et le nombre de clients est représenté par l'ensemble V. Chaque client i a une demande q_i et chaque véhicule disponible dans l'ensemble M à une capacité Q. La formulation utilise une variable c_{ij} qui représente le coût de l'arête entre les sommets i et j, ainsi qu'une variable binaire x_{ijk} qui est égale à 1 si L'arête [i, j] est parcouru par le véhicule k, 0 sinon.

On peut formuler le problème de la manière suivante :

Fonction-Objectif :

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} \sum_{k=1}^{M} x_{ijk} \tag{2.6}$$

Contraintes

$$\sum_{i=1}^{N} \sum_{k=1}^{M} x_{ijk} = 1 \qquad \forall 1 \leq j \leq N \tag{2.7}$$

$$\sum_{i=1}^{N} \sum_{k=1}^{M} x_{ijk} = 1 \qquad \forall 1 \leq i \leq N \tag{2.8}$$

$$\sum_{j=1}^{N} \sum_{l=1}^{N} x_{ilk} = \sum_{l=1}^{N} \sum_{j=1}^{N} x_{ljk} \tag{2.9}$$

$$\sum_{i=1}^{N} x_{i0k} = 1 \qquad \forall 1 \leq k \leq M \tag{2.10}$$

$$\sum_{i=1}^{N} \sum_{j=1}^{N} x_{ijk} \leq Q \qquad \forall 1 \leq k \leq M \tag{2.11}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall 1 \leq k \leq M \tag{2.12}$$

La fonction objectif (2.6) vise à minimiser la somme des coûts de toutes les tournées. Les contraintes (2.7) et (2.8) assurent que chaque client doit être desservi une et une seule fois et les contraintes (2.9) assurent que les flots sont conservés. Les contraintes (2.10) assurent que chaque tournée commence et se termine au dépôt. Finalement, les contraintes (2.11) sont les contraintes de capacité et les contraintes (2.12) mettent en place une restriction binaire sur la variable de décision x_{ijk}.

2.3 Méthode de programmation linéaire

Définition La programmation linéaire (PL) est une technique mathématique d'optimisation qui consiste à modéliser un problème de recherche opérationnelle (RO) où la fonction objectif et les contraintes ont toutes la forme linéaire. Cette modélisation nécessite l'identification de la fonction objectif dont le but est de minimiser (ou de maximiser) celle-ci ainsi que les variables de décision et les différentes contraintes auxquelles sont soumises ces variables. On parle de la programmation linéaire mixte (MIP) si on utilise à la fois des variables entières et non entières.

2.4 Méthodes de résolution

Pour la résolution des problèmes de recherche opérationnelle, des nombreux efforts ont été faits afin de développer une méthode qui permet d'obtenir une solution optimale. On peut distinguer deux types de méthodes de résolution :

- **Les méthodes exactes** : elles permettent l'obtention des solutions optimales pour le problème posé mais pas dans un temps polynomiale. Parmi les méthodes les plus utilisées, on peut citer : les méthodes de génération de colonnes (Annexe :A.1), les méthodes de coupes (Annexe :A.2) et les méthodes de la programmation dynamique (Annexe :A.3).
- **Les méthodes approchées** : elles permettent de trouver des solution dans un temps raisonnable sans pouvoir affirmer qu'elles sont optimales. Parmi les méthodes les plus utilisées, on peut citer : les colonies de fourmis (Annexe :B.1), La recherche tabou (Annexe :B.2), l'algorithme génétique (Annexe :B.3) et le recuit simulé.

La la figure 2.3 résume les différentes méthodes de résolution :

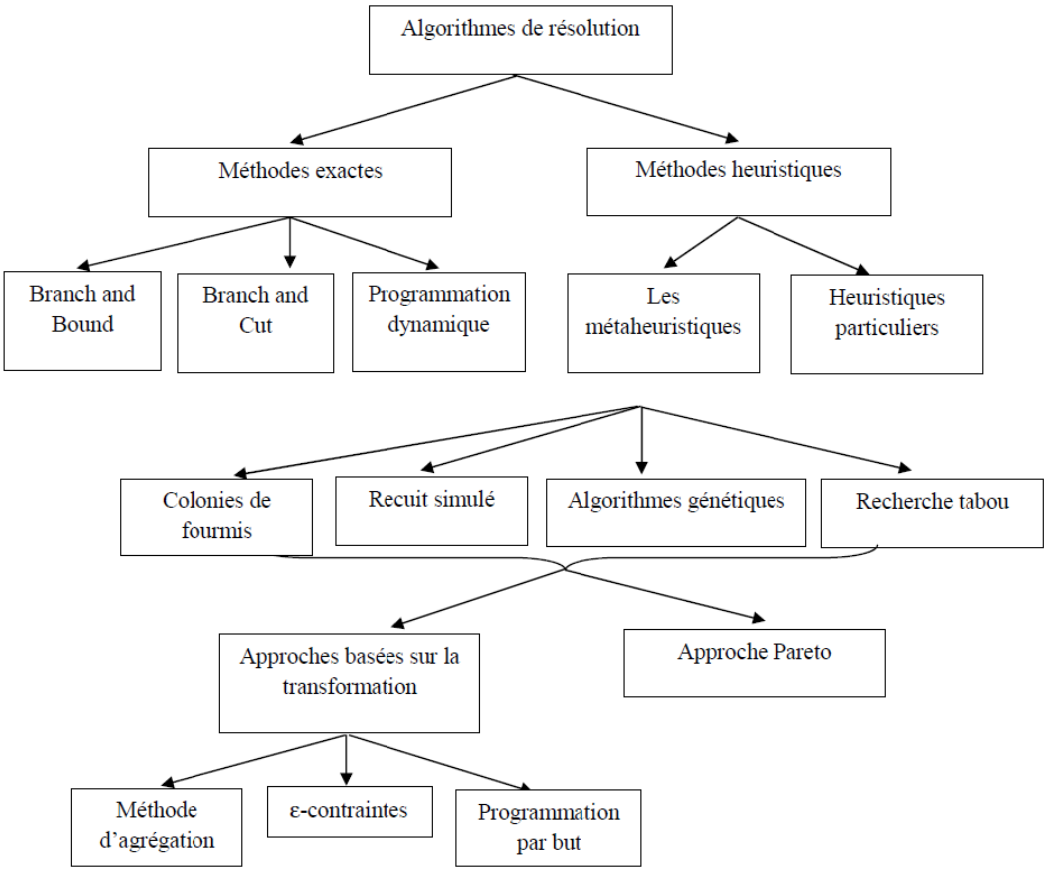

FIGURE 2.3 – Classification des méthodes de résolution [2]

2.5 Problème de collecte de déchets

Bien que la problématique de l'optimisation de la collecte des déchets post-inondation ait été très peu étudiée, le cas général de celle-ci "l'optimisation de la collecte des déchets" l'a beaucoup été.

Au niveau de ces études, le problème de gestion de collecte de déchets à été modélisé comme étant un problème de VRP dont l'objectif principale est soit de minimiser le temps total de collecte ou la distance totale parcourue, soit de minimiser le nombre de véhicules nécessaires à cette collecte [13].

La plupart des recherches ont porté sur des problèmes réels. Ong et al [14], en 1990, ont étudié les circuits de collecte à Singapour. Ils ont modélisé le problème comme étant un VRP dont la fonction objectif est la minimisation de la distance totale parcourue. Pour la résolution de ce problème, ils ont proposé une méthode

de résolution basée sur une heuristique route-first-cluster-second. Cette heuristique comporte deux phases. La première phase consiste à construire une tournée géante de TSP, sans tenir compte les contraintes de capacité des véhicules. La deuxième phase consiste à décomposer cette tournée en des tournées faisables.

En 2000, Mourao et al [15] ont modélisé le problème de collecte des déchets comme étant un problème de tournées sur les arcs qui consiste à créer une tournée géante dans un premier temps puis la décomposer avec des méthodes de bornes inférieures en plusieurs tournées tout en prenant compte la capacité du camion. Pour la résolution de ce problème, une heuristique de route-first-cluster-second a été utilisée.

Shih et al [16] ont étudié le problème de traitement des déchets infectés hospitaliers. Ils ont modélisés le problème comme étant un problème de tournées de véhicules périodique dont l'objectif est la minimisation du coût. Finalement, pour la résolution du problème, ils ont développé un algorithme en deux phases : une première phase consiste à créer des circuits de collecte en utilisant une méthode de programmation dynamique et une seconde phase qui consiste à affecter les tournées aux jours dans la semaine.

Labadi [17], en 2003, a proposé un algorithme génétique pour la résolution du problème de collecte de déchets sur les arcs avec une demande variable selon les jours. Minciardi et al [18] ont étudié le problème de la collecte des déchets en Italie. Ils ont modélisé le problème comme étant un CVRP dont la fonction objectif est la minimisation du temps total des tournées. Ils ont proposés un algorithme de construction de solution basée sur des quartiers pour la résolution du problème.

En 2006, Lacomme et al [19], ont proposé une modélisation sur les arcs (Arc Routing Problem) pour le problème de collecte de déchets pour la ville de Troyes dont l'objectif est de minimiser les distances de parcours et la plus grande tournée. Pour la résolution de ce problème, ces derniers ont proposé un algorithme génétique multi-objectif (la minimisation de la durée totale de l'ensemble des tournées ainsi que la minimisation de la durée de la plus longue tournée effectuée.)

Kim et al [20], en 2007, ont étudié le problème de collecte de déchets à Seoul (Corée du Sud). Ils ont traité le problème selon deux façons : une déterministe et une stochastique ayant le même objectif qui est la minimisation des coûts.

Bautista et al [21], en 2008, ont poursuit une étude sur la gestion de collecte des déchets dans un quartier de Barcelone (Espagne) débutée en 2004 dont l'objectif est

la minimisation des coûts. Ils se sont basés sur l'algorithme des colonies de fourmis pour la résolution du problème.

Dans l'ensemble des études réalisées, la modélisation du problème de gestion de collecte des déchets à porté principalement sur des modélisations de type VRP et l'objectif dominant était la minimisation du coût.

2.6 Conclusion

Lors d'une inondation, plusieurs questions liées à la localisation des entrepôts et à l'optimisation des tournées de ramassage se posent pour mieux gérer la collecte de la quantité importante de déchets produite. Dans la littérature, cette problématique n'a pas été spécifiquement traité mais se rattache à un champs plus large qui est celui de la collecte des déchets.

La différence entre ces deux problèmes est que dans les problèmes de collecte des déchets, les centres de stockages sont connus alors que pour les problèmes de collecte des déchets post-inondation une localisation des dépôts de stockage doit être faite. Durant ce chapitre, nous avons présenté un état de l'art sur ces problèmes qui ont été traités principalement comme étant des problèmes de VRP.

Chapitre 3

Modélisation du problème et méthodes de résolution

3.1 Introduction

Comme il a déjà été mentionné dans les deux chapitres précédents, il est important de s'intéresser aux problèmes de collecte de déchets post-inondation qui n'ont pas été spécifiquement traité. Dans ce chapitre, une description du problème de collecte de déchets post-inondation est proposée. Bien que les objectifs classiques cherchent généralement à minimiser les coûts de transport, le plus important lors d'une inondation semble plutôt la minimisation du temps de collecte. En se basant sur cet objectif, nous avons premièrement travaillé sur une méthode exacte basée sur un modèle de programmation linéaire mixte (MIP) pour valider les formulations sur des petites instances. Ensuite, pour la résolution des grandes instances, nous sommes fondés une méthode approchée.

L'approche proposée comporte deux niveaux. Un premier niveau qui cherche à optimiser les localisations des centres de stockage des déchets. Ce niveau se rattache aux problèmes de localisation-allocation. En effet, contrairement à une gestion classique des déchets, la localisation des centres de stockages et leur dimensionnement reste une question ouverte dans le cadre d'une inondation. A partir de ces résultats, le deuxième niveau cherche alors à optimiser la tournée des véhicules de collecte pour un centre de stockage donné.

3.2 Description du problème

Le but de cette étude est de trouver une meilleur stratégie de collecte de déchets qui permet de satisfaire la totalité des demandes des points de collecte tout en minimisant les coûts et le temps de réponse.

Notre problème peut être défini sur un graphe non orienté complet
G = (V,E) où $V = \{0, 1, .., n\}$ est l'ensemble de sommets qui représentent les points de collecte et les dépôts de stockage, $E = \{[i, j] : i, j \in V, i < j\}$ est l'ensemble des arêtes qui représentent le réseau routier. Chaque arête [i, j] a un coût t_{ij} qui représente le temps de parcours entre le dépôt de stockage i et le point de collecte j, et chaque point de collecte a une demande Q_j qui doit être satisfaite.

Durant la suite du notre mémoire nous allons considérer principalement les données d'entrée suivantes :

N : le nombre de points de collectes ;
M : le nombre des dépôts de stockage ;
Q_j : la quantité à collecté sur un point de collecte ;
C_i : la capacité maximale d'un dépôt de stockage ;
T_{ij} : la matrice de temps de parcours entre un point de collecte et un dépôt de stockage.

3.3 Modélisation du problème de collecte de déchets post-inondation sous la forme MIP

Peu de temps après une inondation, les questions d'optimisation qui se posent sont liées principalement à la localisation des centres de stockages (les installations qui ont pour mission le stockage des déchets) et l'affectation correspondante des points de collecte (points de regroupements provisoires des déchets).

Ainsi pour répondre à ces questions, nous proposons trois modèles :

- Modèle 1 : consiste à localiser les dépôts de stockages tout en minimisant les nuisances engendrés par ces centres de stockages pour le voisinage : Problème de localisation (LP).
- Modèle 2 : consiste à affecter de manière optimale les déchets aux centres de stockage en tenant compte des capacités de stockage : Problème d'allocation (AP).
- Modèle 3 : qui contraint davantage le modèle 2 en cherchant à minimiser le nombre d'allers-retours : Problème de VRP.

Durant les sections suivantes chaque modèle a été traité séparément.

3.3.1 Problème de localisation

Formulation mathématique

Pour localiser les dépôts de stockage dont l'emplacement est inconnu a postériori, nous avons proposé une formulation mathématique sous la forme d'un programme linéaire mixte :

Données d'entrée :

Q_{jt} : La quantité à collecté sur un point de collecte à une période t

N_i : Variable de nuisance : la mauvaise localisation des centres de stockages peuvent constituer une gêne pour la santé des habitants (les mauvaises odeurs, les maladies..).

Les variables de décision :

$$O_{it} = \begin{cases} 1 & \text{si un dépôt de stockage i est ouvert à une période t} \\ 0 & \text{sinon.} \end{cases}$$

$$Y_{ijt} = \begin{cases} 1 & \text{si le point de collecte j est affecté au dépôt de stockage i à une période t} \\ 0 & \text{sinon.} \end{cases}$$

Fonction-Objectif :

$$\min \sum_{i=1}^{M} O_{it} * N_i \tag{3.1}$$

Contraintes

$$\sum_{i=1}^{M} Y_{ijt} \leq 1 \qquad \forall j \in 0..N, \forall t \tag{3.2}$$

$$\sum_{j=1}^{N} Y_{ijt} \leq O_{it} \qquad \forall i \in 0..M, \forall t \tag{3.3}$$

$$\sum_{j=1}^{N} Q_{jt} * Y_{ijt} \leq C_i \qquad \forall i \in 0..M, \forall t \tag{3.4}$$

$$O_{it} \in \{0, 1\} \tag{3.5}$$

$$Y_{ijt} \in \{0, 1\} \tag{3.6}$$

L'équation (3.1) est la fonction objectif qui sert à minimiser le nombre des dépôts à ouvrir à une période de temps t en fonction de la variable de nuisance. La contraintes (3.2) assure qu'à chaque période de temps un point de collecte n'est affecté qu'à un seul dépôt de stockage. La contrainte (3.3) assure que le point de collecte n'est affecté qu'au dépôt de stockage ouvert. La contrainte (3.4) assure que la quantité des déchets collectée ne doit pas dépasser la capacité de dépôt. Les contraintes (3.5) et (3.6) mettent en place une restriction binaire sur les variables.

3.3.2 Problème d'affectation

En reprenant les contraintes (3.2), (3.4) et (3.6), la formulation mathématique sous la forme d'un programme linéaire mixte (MIP) du notre problème d'affectation est la suivante :

Les variables de décision :

$$Y_{ij} = \begin{cases} 1 & \text{si le point de collecte j est affecté au dépôt de stockage j} \\ 0 & \text{sinon.} \end{cases}$$

Fonction-Objectif :

$$\min \sum_{i=1}^{M} \sum_{j=1}^{N} Y_{ij} * T_{ij} \tag{3.7}$$

Contraintes

$$\sum_{i=1}^{M} Y_{ij} \leq 1 \qquad \forall j \in 0..N \tag{3.8}$$

$$\sum_{j=1}^{N} Q_j = \sum_{j=1}^{N} Q_j * Y_{ij} \quad \forall i \in 0..M \tag{3.9}$$

L'équation (3.7) est la fonction objectif qui sert à minimiser le temps de réponse. La contrainte (3.8) assure que la quantité des déchets collectée ne doit pas dépasser la capacité de dépôt. La contrainte (3.9) assure que la totalité des déchets doit être collecté.

3.3.3 Problème d'affectation et minimisation de nombre d'aller et de retour

Dans le but de minimiser le nombre d'allers et de retours entre les centres de stockage et les points de collecte, en utilisant les contraintes (3.2), (3.4), (3.6) et (3.9), une formulation mathématique sous la forme d'un programme linéaire mixte (MIP) du notre problème a été faite.

Les variables de décision :

$F_{ij} \geq 0$: Le nombre d'aller et de retour entre un point de collecte et un dépôt de stockage

Fonction-objectif :

$$\min \sum_{i=1}^{M} \sum_{j=1}^{N} Y_{ij} * T_{ij} + \sum_{i=1}^{M} \sum_{j=1}^{N} F_{ij} * T_{ij} \tag{3.10}$$

Contraintes

$$F_{ij} * Y_{ij} = \frac{Q_j * Y_{ij}}{k_i * C_{k_i}} \quad \forall i \in 0..M, \forall j \in 0..N \tag{3.11}$$

$$F_{ij} \geq 0 \tag{3.12}$$

L'équation (3.10) est la fonction objectif qui sert à minimiser le temps de réponse ainsi que le nombre d'aller et de retour entre le point de collecte et le dépôt de stockage. La contrainte (3.11) assure le calcul de nombre d'aller et de retour. La contrainte (3.12) représente la contrainte de non-négativité sur les variables F_{ij}.

3.3.4 Problème de tournées de véhicules

Pour minimiser les tournées de collecte, notre problème de tournées de véhicules a été formulé mathématiquement sous la forme d'un programme linéaire mixte :

Données d'entrée :

K : Le nombre de véhicules
C : La capacité maximale d'un dépôt de stockage
C_k : La capacité d'un véhicule K
S_k : Le temps d'opération maximum pour un véhicule k

Les variables de décision :

$$Y_{ijk} = \begin{cases} 1 & \text{si l'arc (i,j) est parcourus par k} \\ 0 & \text{sinon.} \end{cases}$$

Fonction-objectif :

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} Y_{ijk} * T_{ij} \tag{3.13}$$

Contraintes

$$\sum_{i=0}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} Q_j * y_{ijk} \leq C_k \tag{3.14}$$

$$\sum_{k=1}^{K} \sum_{j=1}^{N} \sum_{i=0}^{N} Q_j * Y_{ijk} \leq C \tag{3.15}$$

$$\sum_{j=1}^{N} Q_j = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} Q_j * y_{ijk} \tag{3.16}$$

$$\sum_{j=1}^{N} Y_{ijk} * T_{ij} \leq S_k \tag{3.17}$$

$$\sum_{j=1}^{N} Y_{0jk} = 1 \tag{3.18}$$

$$\sum_{j=1}^{N} Y_{j0k} = 1 \tag{3.19}$$

$$\sum_{i=1}^{N} Y_{ilk} = \sum_{j=1}^{N} Y_{ljk} \tag{3.20}$$

$$\sum_{k=1}^{K} Y_{ijk} \leq K \tag{3.21}$$

$$\sum_{i,j}^{S} Y_{ijk}|S| - 1 \quad \forall S \in V, 2 \leq |S| \leq n-2 \quad \forall k \in K \tag{3.22}$$

$$Y_{ijk} \in \{0, 1\} \tag{3.23}$$

L'équation (3.13) est la fonction objectif qui sert à minimiser le temps de réponse. La contrainte (3.14) assure que la quantité des déchets collectée ne doit pas dépasser la capacité de véhicules. La contrainte (3.15) assure que la quantité des déchets collectée ne doit pas dépasser la capacité de dépôt. La contrainte (3.16) assure que la totalité des déchets doit être collecté. La contrainte (3.17) assure que la durée de tournée d'un véhicule ne doit pas dépasser le temps d'opération d'un véhicule. Les contrainte (3.18) et (3.19) signifient que chaque tournée commence et se termine au dépôt. La contrainte (3.20) garantie que chaque point de collecte visité doit être quitté. La contrainte (3.21) assure que le nombre des tournées est limité au nombre des véhicules disponibles au dépôt. La contrainte (3.22) évite la formation des sous-tours et la contrainte (3.23) met en place une restriction binaire sur les variables.

3.3.5 Tests et limites

Les formulations MIP ont été résolues à l'aide du ILOG CPLEX version 12.5 d'IBM qui a été utilisée comme le solveur de programmation mixte et en nombres entiers. Pour l'exécution de nos tests, une machine , sur laquelle un système d'exploitation Windows7 est installé, a été utilisée. Cette machine est caractérisée par une mémoire 4G, un processeur Intel Core i7, et une fréquence de 2,70 GHz, (Annexe :C).

Tests pour le problème d'affectation :

Pour tester notre modèle, on a utilisé des données qui sont générées aléatoirement en accord avec les données que pouvaient fournir la méthode MECADEPI. Le tableau ci dessous représente les valeurs obtenues par le modèle de programmation linéaire mixte (MIP) :

NB dépôts de stockage	NB points de collecte	Fonction objectif	Temps
4	9	3.4	3.2
5	10	4.123	6.2
5	15	5.2	9.2
15	15	6.678	1200
15	20	7	1400
15	25	-	-

TABLE 3.1 – Résultats pour le problème d'affectation sous MIP

Les résultats obtenus au niveau du tableau 3.1 montrent que le temps nécessaire pour résoudre le problème d'affectation croît de façon exponentielle avec l'augmentation du nombre de points de collecte. Ainsi, pour des instances avec un nombre dépôts de stockage égale à 5, on obtient des solutions optimales dont le temps d'exécution est de moins de 10 secondes. Alors que, pour des instances avec un nombre dépôts de stockage égale à 15, MIP donne des solutions faisables avec un temps d'exécution de plus de 1000 secondes.

Pour l'instance avec un nombre de points de collecte égale à 15 et le nombre de points de collecte est égale à 25 MIP ne donne pas de résultat. Ces résultats montrent les limites de MIP face aux instances de grandes tailles d'où la nécessité de s'intéresser aux méthodes approchées.

Tests pour le problème de tournées de véhicules :

Pour tester notre modèle, on a utilisé des données qui sont générées aléatoirement. Le tableau ci dessous représente les valeurs obtenues par le modèle de programmation linéaire mixte :

Les résultats obtenus au niveau du tableau 3.2 montrent que le temps nécessaire pour résoudre le problème de VRP de façon exponentielle avec l'augmentation du nombre de points de collecte. Ainsi, pour des instances ayant un seul dépôt de stockage et un nombre de véhicules qui est égale à 3, on obtient des solutions optimales dont le temps d'exécution qui ne dépasse pas les 10 secondes.

NB dépôts de stockage	NB points de collecte	NB véhicules	Fonction objectif	Temps
1	3	3	7	1.27
1	6	3	8.453	5.09
1	9	3	7.654	10.56
3	10	4	6.876	900
3	20	4	9.56	1800
3	30	4	-	-

TABLE 3.2 – Résultats pour le problème de tournées de véhicules sous MIP

Alors que, pour des instances avec un nombre dépôts de stockage égale 3, et un nombre de véhicules qui est égale à 4, le MIP donne des solutions faisables avec un temps d'exécution de plus de 1000 secondes. Finalement, pour l'instance ayant un nombre de points de collecte égale à 30, le MIP ne donne pas de résultat.

Alors, pour le problème de VRP, le MIP ne peut résoudre avec les moyens techniques à l'optimum que des instances de petites tailles, d'où la nécessité de s'intéresser aux méthodes approchées pour simuler des données réelles pas encore produites par la méthode MECADEPI.

Dans la section suivante une métaheuristique a été proposée pour la résolution du problème de collecte de déchets post-inondation.

3.4 Description de méthode approchée de résolution

Au niveau de nombreuses recherches effectuées dans l'étude des problèmes de collecte de déchets, plusieurs méthodes de résolution qui permettent d'obtenir une solution optimale ont été proposées pour la résolution de ce problème. Ces méthodes de résolutions sont basées principalement sur les algorithmes de colonies de fourmis, le recherche tabou et les algorithmes génétiques,(voir 1.3).

Le recuit simulé est une métaheuristique qui a été développée par Kirkpatrick et al en 1983 [22] et inspirée d'un processus utilisé en métallurgie alternant des des cycles de refroidissement lent et de réchauffage dont l'objective est la minimisation de l'énergie du matériau.

Dans cette méthode le paramètre de contrôle de l'optimisation joue le rôle de la température et l'effet de La température du système à optimiser doit être similaire à celle de la température du système physique. Le but de cette métaheuristique est d'éviter les minima locaux.

3.4.1 Recuit simulé pour le problème de collecte de déchets post-inondation

Dans cette section nous allons proposer une méthode de résolution basée sur le recuit simulé pour résoudre le problème de collecte de déchets post-inondation.

Recuit Simulé pour le problème d'affectation L'algorithme présenté ci-dessous décrit notre algorithme de recuit simulé d'affectation.

Algorithm 1 Recuit Simulé d'affectation

$S \Leftarrow solutionInitialisation$
$T \Leftarrow T_0$
for $i := 0 \rightarrow N$ **do**
  $S' \Leftarrow Affectation()$
  if $(f(S') < f(S)$ **then**
    $S \Leftarrow S'$
  else if $Random[0,1] < \exp \frac{-\Delta t}{T}$ **then**
    $S \Leftarrow S'$
  end if
end for
Reduce the temperature

Cette algorithme commence par l'initialisation d'une solution initiale faisable S, qui représente l'affectation aléatoire des points de collecte aux centres de stockage, et l'initialisation de la température T par une température initiale T_0 qui est assez élevée. A chaque itération de l'algorithme une modification de la solution basée sur l'algorithme de 2-opt est effectuée. Cette modification consiste à échanger a chaque fois entre deux centres de stockages. Cette modification entraine une variation de l'énergie de système qui représente la minimisation des distances entre les points de collecte et les dépôts de stockage tout en tenant compte les contraintes de capacité des dépôts de stockage.

Si le coût de cette solution est inférieur au coût de la solution courante S alors l'algorithme conserve la nouvelle solution S'. Sinon, il choisi un nombre aléatoire entre 0 et 1 et il le compare avec la critère d'acceptation $\exp \frac{-\Delta t}{T}$, si ce nombre est inférieur alors il garde cette solution. Sinon, S' sera rejetée. Finalement, après N itérations, il diminue la température T.

Recuit Simulé pour le problème de VRP En se basant sur les résultats d'affectation obtenus par l'application de l'algorithme de recuit Simulé pour le problème d'affectation, l'algorithme de recuit Simulé pour le problème de VRP a été élaboré.

Il commence par l'initialisation d'une solution initiale faisable S, qui représente l'élaboration des routes entre le dépôt de stockage et les points de collecte, et l'initialisation de la température T par une température initiale T_0 qui est assez élevée. A chaque itération de l'algorithme une modification de la solution basée sur l'algorithme de 2-opt est effectuée.

Cette modification entraine une variation de l'énergie de système qui représente la minimisation de temps de parcours des points de collecte tout en respectant le temps d'opération des véhicules. Si le coût de cette solution est inférieur au coût de la solution courante S alors l'algorithme conserve la nouvelle solution S'. Sinon ,il choisi un nombre aléatoire entre 0 et 1 et il le compare avec la critère d'acceptation $\exp \frac{-\Delta t}{T}$, si ce nombre est inférieur alors il garde cette solution. Sinon, S' sera rejetée. Finalement, après N itérations, il diminue la température T.

Algorithme 2-opt Il s'agit d'un algorithme de recherche locale itératif. Cet algorithme permet d'améliorer une solution initiale. Il consiste à supprimer à chaque étape, deux arêtes de la solution courante et insérer deux autres arêtes pour compléter les routes. La figure 3.1 présente l'exécution de cet algorithme

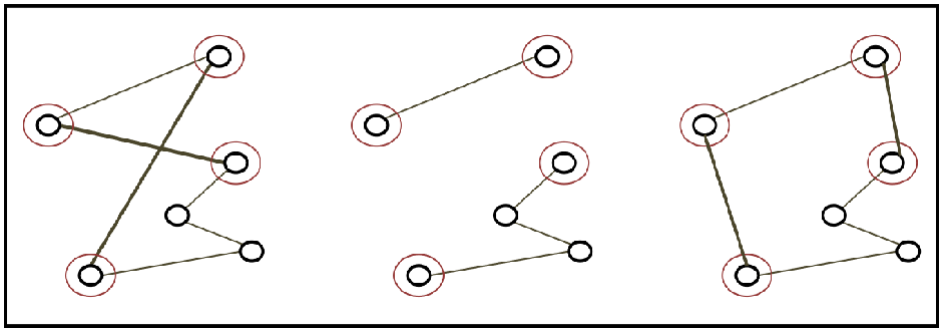

FIGURE 3.1 – Exécution de l'algorithme de 2-opt

3.5 Conclusion

Dans ce chapitre, une description ainsi qu'une formulation des problématiques ont été faites. Nous avons proposé deux méthodes de résolution : une méthode exacte et une méthode approchée. Pour la méthode exacte, il s'agit d'un modèle de programmation linéaire mixte. Ce modèle vise à minimiser le temps de réponse et satisfaire la totalité de la demande des points de collecte. Pour la méthode approchée, il s'agit d'une métaheuristique basée sur le recuit simulé.

Conclusion et perspectives

Les inondations constituent la première menace naturelle dans le monde entier, elles génèrent des quantités très importants des déchets. La collecte de ces déchets est la première étape à faire pour reprendre les activités économiques et sociales des territoires. Malgré l'importance et la sensibilité de cette problématique, elle n'a pas été beaucoup traitée.

Dans ce mémoire, notre étude porte sur le problème de gestion de collecte de déchets post-inondation. Notre objectif est de minimiser le temps de réponse ainsi que de satisfaire la totalité de la demande des points de collectes tout en tenant compte les contraintes de capacité des dépôts de stockage.

Dans ce cadre, après un chapitre introductif et un état de l'art sur la question, nous avons proposés dans le troisième chapitre une modélisation du problème. Il s'agit d'une modélisation multi-échelle s'appuyant sur une première modélisation de type localisation-allocation, et d'une deuxième plus fine de type VRP.

Ces modèles ont été formulés sous la forme d'un MIP puis à l'aide d'un jeu test, il s'est avéré nécessaire de résoudre ces modèles par des métaheuristiques. Ainsi nous avons proposé une méthode approchée basée sur le recuit simulé pour la résolution des instances de grandes tailles. Finalement, cet algorithme sera appliqué sur le territoire francilien pour répondre à la question de départ.

Les modèles et les algorithmes de résolution (encore en cours de développement) seront appliqués au territoire francilien. En effet, le premier axe du projet SIGOPT a développé une extension à un SIG libre qui permet d'obtenir les quantités de déchets générés par une inondation de type 1910. A partir de ces valeurs, nous pourrons prochainement contribuer à localiser de manière optimale les centres de stockage des déchets et organiser les tournées de véhicule.

Bibliographie

[1] K.Julien. Paris : la crue de 1910 rencontre celle de juin 2016, 2016.

[2] H.D.Imen. *Optimisation heuristique pour la résolution du m-PDPTW statique et dynamique*. PhD thesis, Ecole Centrale de Lille, Ecole Nationale d'Ingénieurs de Tunis (Tunisie), 2010.

[3] CEPRI. Gestion des déchets post-inondation, approche pour une méthodologie d'élaboration de plans de gestion. Technical report, 2012.

[4] MEDDE. Mieux savoir pour mieux agir. principaux enseignements de la première évaluation des risques d'inondation sur le territoire français. Technical report, 2012.

[5] J.Bonnemains. Les déchets post catastrophe. anticiper pour mieux gérer. *TSM*, pages 60–69, 2009.

[6] Robin des Bois. Compte rendu groupe de travail déchets post-catastrophe. 2008.

[7] R.F.Love J.G.Morris, G. Wesolowsky. *Facilities Location : Models and Methods*. Operations Research Series, 1988.

[8] et T. RAY M.EFROYMSON. A branch-bound algorithm for plant location. *Operations Research*, 1966.

[9] E.L. Lawler A.H.G.R. Kan, J.K. Lenstra and D.B. Shmoys. The traveling salesman problem : a guided tour of combinatorial optimization. *Wiley New York*, 1985.

[10] G.B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 1959.

[11] H. Dickhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 1990.

[12] F. Semet and T.G. Crainic. echerche opérationnelle et transport de marchandises. in optimisation combinatoire. *Hermès Science :Lavoisier*, 2006.

[13] A. Runka B.M. Ombuki-Berman and F.T. Hanshar. aste collection vehicle routing problem with time windows using multi-objective genetic algorithms. Technical report, 2007.

[14] H.L. Ong T.N. Goh and K.L. Poh. A. A computerised vehicle routing system for refuse collection. *Advances in Engineering Software*, 1990.

[15] M.C. Mourao and M.T. Almeida. Lower-bounding and heuristic methods for a refuse collection vehicle routing problem. *European Journal of Operational Research*, 2000.

[16] L.H. Shih and H.C. Chang. A routing and scheduling system for infectious waste collection. *Environmental Modeling and Assessment*, 2001.

[17] N. Labadi. *Problèmes d'optimisation en tournées sur les arcs*. PhD thesis, Université de technologie de Troyes, 2003.

[18] R. Minciardi M. Paolucci and E. Trasforini. A new procedure to plan routing and scheduling of vehicles for solid waste collection at a metropolitan scale. *Presented at Odysseys, Palermo, Italy,*, 2003.

[19] P. Lacomme C. Prins and M. Sevaux. A genetic algorithm for a bi-objective capacited arc routing problem. *Computers and Operations Research*, 2006.

[20] J.D. Kim H.S. Choi and D.H. Lee. Vehicle routing in a refuse collection system : a case study. *Computational Science and its Applications, ICCSA 2007*, 2007.

[21] J. Bautista E. Fernández and J. Pereira. Solving an urban waste collection problem using ants heuristics. *Computers and Operations Research*, 2008.

[22] C. D. Gelatt Kirkpatrick and M. P. Vecchi. Optimization by simulated annealing. *Science*, 1983.

[23] E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 1958.

[24] P.Borne F.Rotella I.Zambettakis G.D Tanguy, J.P Richard. *Automatique commande et optimisation des processus*. 1990.

[25] A. Colorniand V. Maniezzo M. Dorigo. Distributed optimization by ant coloniess. *Proceeding of the first European Conference on Artificial Life (ECAL 91)*, 1992.

[26] L.M. Gambardella E.Lucibello A.V.Donati A.E Rizzoli, F.Oliverio and R.Montemanni. Ant colony optimization for vehicle routing in advanced logistics systems. *IDSIA, Galleria 2, 6928 Manno, Switzerland and AntOptima, via Fusoni 4, 6900 Lugano, Switzerland*, 2003.

[27] F.Glover. Tabu search : part i. *ORSA Journal of Computing*, 1989.

[28] J. Holland. Adaptation in natural and artificial systems. *University of Michigan Press, Ann Arbor, Canberra ACT 2601.Autralia*, 1975.

Annexe

A Les méthodes exactes

A.1 Les méthodes de génération de colonnes

La génération de colonnes est une ancienne méthode utilisée pour résoudre un problème de programmation linéaires ayant une grande taille. Cette méthode sert à résoudre un problème réduit appelé restreint ayant un ensemble limité de variables dont le but d'avoir une solution optimale au problème linéaire initial qui est le problème maître. pour atteindre la solution optimale du problème maître, si l'ensemble des variables du problème restreint ne contient pas celles qui permettent l'obtention de la solution optimale pour le problème initial, l'ajout des variables pouvant être utilisées pour améliorer la solution au problème restreint est obligatoire.

A.2 Les méthodes de coupes

Les méthodes de coupes appartiennent à la classe des méthodes itératives. Ces méthodes peuvent être résumé en trois étapes :

- La résolution du programme linéaire (PL) sans tenant compte les contraintes d'intégrité sur les variables(simplexe). Cette résolution entraine l'obtention d'une solution optimale qui n'est pas nécessairement entière (si la solution est entière alors elle s'arrête).
- L'élimination de la solution non-entière obtenue précédemment tout en rajoutant une contrainte supplémentaire qui s'appelle une coupe.
- La répétition de la dernière étape jusqu'à l'obtention d'une solution entière.

Parmi les méthodes des coupes les plus connus est la coupe de Gomory qui a été introduite en 1958 [23]. Le but de cette méthode est de remplacer les contraintes dans le PLNE avec des autres contraintes linéaires qui représentent les coupes.

A.3 Programmation dynamique

La programmation dynamique est une méthode efficace proposée par le mathématicien Richard Bellman en 1954 pour la résolution des problèmes de chemins optimaux :" Si C est un point qui appartient au chemin optimal entre A et B, alors la portion de ce même chemin allant de A à C est le chemin optimal entre A et C, [24]. Cela signifie que pour l'obtention du chemin optimal, il suffit juste de combiner les différents sous chemins optimaux.

B Les méthodes approchées

B.1 Les colonies de fourmis

En se basant sur le comportement réel et la communication chez les fourmis qui consiste en « phéromone », Coloroni et al [25] ont introduit et appliqué une nouvelle métaheuristique pour la résolution du problème de voyageur de commerce en 1992. Cette méthode a été applique pour la résolution du VRP par Gambardella et al en 2003 [26].

B.2 La recherche tabou

La recherche tabou est une technique de recherche locale, elle a été introduite en 1986 par Glover [27]. En effet, si on sélectionne une solution initiale s qui appartient à l'ensemble de solutions S et qui est engendrée aléatoirement, un sous-ensemble de solution N(s) qui appartient au voisinage S sera générée. La meilleure solution s' sera choisi à partir de l'ensemble de solutions voisines N(s) en utilisant une fonction d'évaluation qui permet de retenir le voisin qui améliore le valeur de f.

B.3 L'algorithme génétique

Les algorithmes génétiques ont été introduites par holland en 1975 [28]. Ces algorithmes sont appelés des algorithmes évolutionnaires puisqu'ils sont inspirés de l'évolution naturelle des espèces. Ces algorithmes se basent tous sur l'évolution d'une population de solutions.Ces solutions, en appliquant certaines règles optimisent un comportement précis, exprimé sous forme d'une fonction coût qui caractérise l'adaptation à l'environnement.

C Code Cplex

```
using CP;
//Data


int nbcust=...;//nombre des points de collecte
int nbdep=...;// nombre des dépôts de stockage
int time=...;
range temps =1..time;
assert( nbcust > nbdep );
range depots=1..nbdep;
range clients=1..nbcust;
//int capacity[depots]=...;
float nuis[depots]=...;
float demand[clients][temps]=...;
float Capacity[depots]=...;
//int T[depots][clients]=...;

// decision variables

//dvar int y[depots][clients];
dvar int O[depots][temps] in 0..1;
dvar int y[depots][clients][temps]in 0..1;

//Objectif function

minimize sum ( i in depots )sum ( t in temps)O[i][t]*nuis[i];

//contraintes
subject to{
  forall (j in clients) forall(t in temps)
   sum ( i in depots) y[i][j][t]==1;

forall( i in depots) forall(t in temps)
forall( j in clients) y[i][j][t] <= O[i][t];
 forall( i in depots)forall(t in temps)
 sum( j in clients) y[i][j][t] * demand[j][t] <= Capacity[i];




}


 main {
  thisOplModel.generate();
   cp.solve();
  var ofile = new IloOplOutputFile("LocationProblem.txt");
  ofile.writeln(thisOplModel.printExternalData());
  ofile.writeln(thisOplModel.printInternalData());
  ofile.writeln(thisOplModel.printSolution());
  ofile.close();
```

FIGURE 2 – Problème de localisation sous Cplex

```
using CP;




int nbCust  = ...; // # of customers
int  nbdept = ...;// # of depots


range  customers  = 1..nbCust;
range   depots = 1..nbdept;


int demand[customers] = ...;
int capacity[depots] = ...;

tuple arc{
int i;
int j;
}
setof(arc) arcs      = {<i,j> | i in depots,j in customers};
float T[arcs] = ...;


dvar int y[depots][customers] in 0..1;

//MINIMISATION DE TEMPS DE REPONSE

minimize( sum ( j in customers) sum ( i in depots ) y[i][j]*T[<i,j>]);


subject to {
//chaque point de collecte ne doit pas être qu'à un seule dépôt
  forall (j in customers )
 sum (i in depots) (y[i][j]) ==1;
   // la quantité collectée ne doit pas dépasser la capacité des dépôts
   forall(i in depots )
      sum(j in customers)(demand[j]*y[i][j])<=capacity[i];
         // la totalité des déchets doit être collecté

forall(i in depots )
sum(j in customers)(demand[j]*y[i][j]) <= sum(j in customers) demand[j];

}
main {
  thisOplModel.generate();
   cp.solve();
  var ofile = new IloOplOutputFile("Affectation.txt");
  ofile.writeln(thisOplModel.printExternalData());
  ofile.writeln(thisOplModel.printInternalData());
  ofile.writeln(thisOplModel.printSolution());
  ofile.close();
```

FIGURE 3 – Problème d'affectation sous Cplex

```
using CP;


int nbCust  = ...; // # of customers
int  nbdept = ...;// # of depots



range customers  = 1..nbCust;
range   depots = 1..nbdept;


float demand[customers] = ...;
float capacity[depots] = ...;
int K[depots]=...;// potentiel de traitement
int c =...;


tuple arc{
int i;
int j;
}
setof(arc) arcs      = {<i,j> | i in depots,j in customers};
int T[arcs] = ...;
int N=1000000;

dvar int y[depots][customers] in 0..1;

dvar int F[depots][customers]in 0..N ;



minimize( sum ( j in customers) sum ( i in depots ) y[i][j]*T[<i,j>])+( sum
( j in customers) sum ( i in depots ) T[<i,j>]*F[i][j]) ;


subject to {
//chaque point de collecte ne doit pas être qu'à un seule dépôt
  forall (j in customers )
 sum (i in depots) (y[i][j]) ==1;
   // la quantité collectée ne doit pas dépasser la capacité des dépôts
   forall(i in depots )
      sum(j in customers)(demand[j]*y[i][j])<=capacity[i];
         // la totalité des déchets doit être collecté

forall(i in depots )
sum(j in customers)(demand[j]*y[i][j]) <= sum(j in customers) demand[j];


forall(i in depots )
sum(j in customers)(demand[j]*y[i][j]) <= sum(j in customers) demand[j];



forall (j in customers )forall (i in depots )
 F[i][j]*y[i][j]== (demand[j]*y[i][j])/(K[i]*c);

}
```

FIGURE 4 – Problème d'affectation avec minimisation de nombre d'aller et de retour sous Cplex

```
using CP;

int n=...;// nombre des sommets

int K=...;

tuple arc{
int i;
int j;
}
setof(arc) arcs = {<i,j> | i,j in 0..n : i!=j};// l'ensemble des arcs

int T[arcs] = ...;

int demand[0..n] = ...;// la quantité des déchets à collecter sur un point
de collecte
int capacity= ...; // la capacitéde dépôt
int c [1..K]=...;// la capacité de véhicule

int t[1..K]=...;// temps d'opération d'un véhicule

int N=10000;

dvar int y [0..n][0..n][1..K] in 0..1; //variable de décision
dvar int Q[0..n]in 0..N; // la quantité totale collectée qui se trouve au
niveau de dépôt


//objective function
//Minimisation de temps de réponse
 minimize  sum( i,j in 0..n : i!=j)sum(k in 1..K) T[<i,j>]*y[i][j][k];

//constraints

subject to{


// un véhicule ne peut affécté qu'à un seule dépôt

forall (ordered i,j in 0..n:i!=j )
sum(k in 1..K) y[i][j][k]<=1;


forall(i in 0..n :i!=0) forall( k in 1..K)
sum(j in 0..n) y[i][j][k]==sum(j in 0..n) y[j][i][k];
//le nombre des véhicules par dépôts =k

 //sum(k in 1..K) y[0][0][k]==K;


//chaque tournée commence et se termine au dépôt

forall (k in 1..K)
sum( j in 1..n) y[0][j][k]==1;
forall (k in 1..K)
sum( j in 1..n) y[j][0][k]==1;
forall (k in 1..K)forall( i in 1..n) y[i][i][k]==0;
```


FIGURE 5 – Problème des tournées des véhicules sous Cplex