Modélisation et résolution de problèmes généralisés de tournées de véhicules

Minh Hoang Ha

l'université nantes angers le mans
PÔLE DE RECHERCHE ET D'ENSEIGNEMENT SUPÉRIEUR

ECOLE DES MINES DE NANTES

Thèse de Doctorat

Minh Hoang HA

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Ecole des Mines de Nantes
sous le label de l'Université de Nantes Angers Le Mans

Discipline : Automatique-Productique
Spécialité : Recherche Opérationnelle
Laboratoire : IRCCyN et CIRRELT

Soutenue le 14 décembre 2012

École doctorale : Sciences et Technologies de l'Information et de Mathématiques (STIM)
Thèse n° : 2012EMNA0068

Modélisation et résolution de problèmes généralisés de tournées de véhicules

JURY

Rapporteurs :	**M. Michel GENDREAU**, Professeur titulaire, Ecole Polytechnique de Montréal/CIRRELT **M. Fédéric SEMET**, Professeur des universités, Université de Lille
Examinateur :	**M. Pierre DEJAX**, Professeur, Ecole des Mines de Nantes/IRCCyN
Directrice de thèse :	**M^{me} Nathalie BOSTEL**, Maître de Conférence HDR, IUT de Saint-Nazaire/IRCCyN
Co-directeurs de thèse :	**M. André LANGEVIN**, Professeur titulaire, Ecole Polytechnique de Montréal/CIRRELT **M. Louis-Martin ROUSSEAU**, Professeur agrégé, Ecole Polytechnique de Montréal/CIRRELT

A toutes les personnes que j'aime.

Remerciements

Mes profonds remerciements s'adressent tout d'abord à mes directeurs de thèse, Nathalie Bostel, André Langevin et Louis-Martin Rousseau, qui ont dirigé mon travail de cherche. Nathalie, je te remercie pour ta confiance, tes encouragements et particulièrement tes nombreuses aides à l'école et même dans la vie. André, merci, pour ton dévouement, ton travail consciencieux et tes aides. Louis-Martin, merci à toi, pour m'avoir fait confiance, pour tes idées pour l'amélioration de mes algorithmes et particulièrement pour m'avoir aidé dans la programmation.

Un très grand merci à Pierre Dejax, mon "quatrième" directeur de thèse, également président de mon jury, pour son aide, ses conseils et son soutien.

Je tiens bien évidement à exprimer ma reconnaissance aux membres du jury : Michel Gendreau et Frédéric Semet, mes rapporteurs, pour avoir relu mon mémoire et jugé mon travail.

Je remercie aussi mes parents, mon frère et ma copine pour leur soutien inconditionnel pendant toutes ces années de thèse.

Finalement, merci à toi, lecteur, de t'intéresser à mon travail.

Table des matières

Liste des tableaux

Table des figures

1

Introduction générale

Cette thèse est consacrée à des applications d'optimisation combinatoire en logistique et transport, et considère quelques problèmes d'optimisation mathématique résolus par la recherche opérationnelle. Le problème étudié dans cette thèse se nomme le problème généralisé de tournées de véhicules. Avant de l'aborder plus spécifiquement, ce chapitre fournit une vue panoramique sur les sujets de recherche principaux, les contributions de la thèse ainsi que le plan de ce mémoire.

Le problème de tournées de véhicules est un des problèmes d'optimisation combinatoire les plus connus et les plus difficiles. Il consiste à déterminer une séquence optimale des clients (problèmes sur nœuds) ou des segments routiers (problèmes sur arcs) pour construire les itinéraires des véhicules. C'est un problème d'optimisation mathématique, c'est-à-dire qu'il est constitué de variables de décision, d'une fonction objectif, et de contraintes à satisfaire. Le développement de technologie appliquée au domaine des transports, la livraison de produits spécifiques ou les nouvelles contraintes concernant la demande des clients génèrent plusieurs nouveaux problèmes de tournées de véhicules, en général plus compliqués et plus difficiles à résoudre. Cette thèse considère une classe nouvelle de problèmes et traite les trois problèmes généralisés de tournées de véhicules suivants :

- Le problème de la tournée sur arcs suffisamment proche (CEARP) (qui généralise le RPP, Rural Postman Problem)
- Le problème de tournées couvrantes multi-véhicules (mCTP) (qui généralise le CVRP, Capacitated Vehicle Routing Problem)
- Le problème généralisé de tournées sur nœuds (GVRP) (qui généralise le CVRP)

Les problèmes sont décrits en détail dans les chapitres respectifs les concernant. Très peu de travaux dans la littérature ont traité ces problèmes. Deux problèmes (CEARP et mCTP) ont été rencontrés dans des entreprises de distribution de l'eau ou d'énergie. Dans ce secteur d'activité, certaines entreprises remplacent les compteurs classiques de mesure de consommation chez les clients par des compteurs "intelligents" pouvant transmettre les informations à distance. Les problèmes CEARP et mCTP correspondent au besoin de créer un itinéraire pour les véhicules qui parcourent les rues pour lire les compteurs en passant à proximité des clients, sans les visiter individuellement.

Que signifie *généralisé* dans ces problèmes ? Habituellement, dans le domaine de l'optimisation mathématique, un problème A est considéré comme une généralisation d'un autre problème B si chaque instance de B constitue un cas particulier de A. Par exemple, le CVRP est une généralisation du TSP parce que chaque instance du TSP peut être considérée comme une instance particulière du CVRP avec un seul véhicule et tous les clients ayant une demande nulle. Un problème de tournées de véhicules est normalement généralisé par l'ajout des contraintes réelles qui se posent dans l'industrie, par exemple : les contraintes de capacité, de fenêtre de temps, etc... Dans le cadre de cette thèse, le mot "généralisé" implique que le problème est étendu par l'ajout de la "contrainte de couverture". Ici, chaque client peut être couvert (ou servi) par un ensemble de nœuds (ou arcs) et il est considéré comme servi si au moins un nœud est visité ou au moins d'un arc est traversé dans cet ensemble. Dans des problèmes de tournées de véhicules classiques, la difficulté consiste à déterminer une séquence optimale de clients ou de segments routiers pour construire des itinéraires respectant l'ensemble des contraintes. Dans les problèmes généralisés, le degré de difficulté augmente car avant de construire des routes, il faut sélectionner des clients ou des segments routiers à visiter.

Cette thèse est une contribution dans le domaine de la recherche opérationnelle (RO) et de l'aide à la décision pour la résolution de problèmes de tournées de véhicules. La RO est un domaine interdisciplinaire qui se situe à la frontière de l'informatique et des mathématiques. La thèse se concentre principalement sur l'aspect mathématique : développement de modèles mathématiques et de méthodes de résolution adaptées. La partie orientée vers la modélisation consiste en formulations de programmation linéaire entière nouvelles, développées pour les trois problèmes étudiés. La partie orientée vers les méthodes consiste au développement algorithmique pour la résolution de ces trois problèmes. Tous les problèmes sont résolus exactement par des algorithmes de branchement et coupes. De plus, les problèmes sont résolus par des méthodes approchées comme des heuristiques constructives basées sur la PLNE et des métaheuristiques pour mieux traiter des instances réelles.

Cette thèse est organisée comme suit. Dans le chapitre 2, nous introduisons les différentes classes de problèmes généralisés de tournées de véhicules. Pour chaque classe, nous présentons sa description, résumons son état de l'art et fournissons des exemples d'applications réelles. Le chapitre 3 est consacré au problème de la tournée sur arcs suffisamment proches (CEARP). Dans les chapitres 4 et 5, le problème de tournées couvrantes multi-véhicules (mCTP) et le problème généralisé de tournées sur nœuds (GVRP) sont traités respectivement. Pour chaque problème, des formulations sont proposées, des algorithmes de résolution sont développés et des résultats d'évaluations numériques sont présentés et analysés. Tous les articles que nous avons publiés ou soumis sont présentés au début de chaque chapitre concerné. Finalement, des conclusions et des propositions de recherches futures sont présentées dans le chapitre 6.

2

Problèmes généralisés de tournées de véhicules

Les problèmes généralisés de tournées de véhicules appartiennent à la classe de problèmes d'optimisation combinatoire généralisés. Les problèmes classiques de tournées de véhicules peuvent être généralisés de façon naturelle en considérant un problème dérivé dans lequel une partition donnée de nœuds (ou arcs) du graphe est regroupée en un ensemble de nœuds (ou arcs), que l'on appellera grappe. Grâce aux nombreuses nouvelles applications récentes, les problèmes généralisés ont un rôle de plus en plus important dans le domaine de la distribution, de la collecte, de la logistique, etc.

Dans ce chapitre, nous présentons les quatre problèmes généralisés de tournées que nous allons traiter dans cette thèse :

- Le problème généralisé de tournées sur nœuds (GVRP ou Generalized Vehicle Routing Problem)
- Le problème de la tournée couvrante (CTP ou Covering Tour Problem)
- Le problème généralisé de tournées sur arcs (GARP ou Generalized Arc Routing Problem)
- Le problème de la tournée sur arcs suffisamment proche (CEARP ou Close Enough Arc Routing Problem)

Pour chacun de ces problèmes, nous décrivons la problématique associée, résumons l'état de l'art et introduisons ses principales applications réelles.

2.1 Problème généralisé de tournées sur nœuds

Avant d'introduire le problème généralisé de tournée sur les nœuds (GVRP), nous abordons d'abord les deux sous-problèmes suivants : le problème généralisé du voyageur de commerce (GTSP) et le problème généralisé du voyageur de commerce avec multiples véhicules (mGTSP).

Le problème généralisé du voyageur de commerce est une généralisation d'un des problèmes les plus connus d'optimisation combinatoire : le problème du voyageur de commerce (TSP). Le GTSP peut être décrit comme suit. Soit $G = (V, E)$

un graphe complet non orienté, $V = \{v_1, ..., v_n\}$ un ensemble de nœuds, E un ensemble d'arêtes et $C = \{C_1, ..., C_K\}$ un ensemble de grappes, avec $0 < K \leq n$. Chaque $v_i \in V$ appartient à exactement une grappe (notons que de cette manière, les grappes sont mutuellement disjointes). Notons c_{ij} le coût (distance, temps) entre les nœuds v_i et v_j. Le but est de chercher un tour de coût minimal visitant chaque grappe exactement une fois.

Le GTSP a été introduit initialement par Srivastava et al. [58] et Henry-Labordere [31]. Dans ces travaux primitifs, le problème a été résolu par la programmation dynamique. Il a ensuite attiré beaucoup d'attention de la communauté de recherche. Plusieurs recherches ont porté sur la transformation du GTSP en TSP (voir par exemple Ben-Arieh et al. [10], Laporte et Semet [39], Noon et Bean [45]). Cette transformation possède deux propriétés importantes : 1) les instances de TSP créées ont presque la même taille que celles du GTSP et 2) une solution optimale du TSP créée peut être convertie en une solution optimale du GTSP. Mais malheureusement, une solution non optimale réalisable du TSP résultant peut être non réalisable pour le GTSP. Par conséquent, la résolution du GTSP demande des solutions optimales des instances du TSP obtenues. Comme les structures des instances du TSP généré et du TSP standard sont différentes, la résolution de manière exacte du GTSP par la transformation en TSP est difficile pour les solveurs actuels du TSP. Une méthode exacte plus efficace pour le GTSP est un algorithme de branchement et coupes développé par Fischetti et al. [21] qui peut résoudre quelques instances jusqu'à 89 grappes et 442 nœuds.

Deux algorithmes d'approximation ont été publiés dans la littérature pour le GTSP. Slavík [56] a proposé un algorithme avec le ratio d'approximation de $3\rho/2$, où ρ est le nombre de nœuds dans la grappe la plus grande ($\rho = \max_{i=1,...,K}(|C_i|)$. Malheureusement, le ratio dans le pire cas peut être relativement mauvais car ρ peut être assez grand. Garg et Konjevod [22] ont proposé un algorithme d'approximation de $\mathcal{O}(\log^2(n)\log(\log(n))\log(m))$. Dans les deux algorithmes, l'inégalité triangulaire doit être satisfaite.

La plupart des travaux dans la littérature traite le problème par une méthode heuristique. Noon et Bean [44] ont proposé quelques heuristiques, incluant une adaptation de l'heuristique du plus proche voisin développée pour le TSP. Des adaptations similaires ont été implémentées par Fischetti et al. [21]. En suite, Renaud et Boctor [51] ont proposé une heuristique appelée GI^3 (Generalized Initialization, Insertion and Improvement en anglais) qui était une généralisation de l'heuristique I^3 présentée dans Renaud et al. [52] pour le TSP. Cette heuristique consiste de trois phases : une phase d'initialisation dans laquelle une partie du tour est construite, une phase d'insertion qui complète le tour en insérant un nœud d'une des grappes non-visitées qui donne le plus d'économie et une phase d'amélioration basée sur les mouvements de 2- et 3-opt. Des heuristiques constructives et des recherches locales ont été aussi discutées dans Bontoux et al. [12], Gutin et Karapetyan [28], Hu et Raidl [33], Snyder et Daskin [57]. Récemment, Karapetyan et Gutin [37] ont donné une révision de voisinages du GTSP et discuté en détail de différentes recherches locales.

Plusieurs métaheuristiques ont aussi été proposées pour le GTSP. Un des algorithmes les plus compétitifs publiés à ce jour est de Silberholz et Golden [54]. Il s'agit d'un algorithme génétique avec quelques nouvelles caractéristiques, incluant des populations initiales isolées et un nouveau mécanisme de reproduction basé sur l'opérateur ordonné de croisement du TSP. Ce nouveau mécanisme a été appelé mrOX (modified rotational ordered crossover en anglais). Les procédures d'amé-

lioration locale combinées avec ce mécanisme permettent d'obtenir de très bons résultats sur de nouvelles instances de grande taille. Plus récemment, Bontoux et al. [12] ont proposé un algorithme mémétique où la procédure de croisement utilise la recherche locale à grand voisinage. Les résultats obtenus sur l'ensemble des instances standards avec jusqu'à 217 grappes et 1084 nœuds ont démontré l'efficacité de l'algorithme en terme de la qualité des solutions et temps de calcul. D'autres métaheuristiques pour le GTSP ont également été développées par Gutin et Karapetyan [28], Gutin et al. [29], Huang et al. [34], Pintea et al. [46], Tasgetiren et al. [60], Snyder et Daskin [57].

Le problème généralisé du voyageur de commerce avec multiples véhicules est une généralisation du problème du voyageur de commerce avec multiples véhicules (mTSP ou multiple Traveling Salesman Problem). Nous n'avons trouvé, à ce jour, aucun article qui traite ce nouveau problème.

Le GmTSP peut être défini comme suit : Soit $G = (V, E)$ un graphe complet non orienté, où V est un ensemble de nœuds et E est un ensemble d'arêtes. $V = \{v_0, ..., v_{n-1}\}$ est l'ensemble de n nœuds qui peuvent être visités, le nœud v_0 est le dépôt où m véhicules de capacité illimitée sont localisés. $C = \{C_0, ..., C_{K-1}\}$ est un ensemble de K grappes disjointes qui doivent être visitées. Chaque nœud $v_i \in V$ appartient à exactement une grappe. Un coût c_{ij} est associé à chaque arête de $E = \{v_i, v_j : v_i, v_j \in V, i < j\}$. L'objectif du GmTSP est de déterminer m tours de coût minimal pour m véhicules, qui commencent et se terminent au dépôt, tels que chaque nœud de $V \setminus \{v_0\}$ soit visité au plus une fois et chaque grappe de C soit visitée exactement une fois.

Le problème généralisé de tournées sur nœuds (GVRP ou Generalized Vehicule Routing Problem) est une généralisation du problème classique CVRP, qui est aussi une extension du GTSP et également du GmTSP.

Etant donné un graphe complet non orienté $G = (V, E)$ et un ensemble de grappes $C = \{C_0, ..., C_{K-1}\}$ définis comme dans le cas du problème GmTSP, les différences par rapport au GmTSP sont les suivantes :

1. au nœud de dépôt v_0, m véhicules identiques sont localisés, d'une capacité commune Q limitée
2. chaque grappe C_i, sauf C_0 (qui se compose uniquement du dépôt), a une demande D_i.

Le GVRP consiste à déterminer m routes telles que :

1. chaque route commence et se termine au dépôt ;
2. exactement un nœud de chaque grappe doit être visité une et une seule fois ;
3. la demande totale de chaque route ne dépasse pas la capacité du véhicule ;
4. le coût total est minimisé.

Le GVRP est clairement NP-difficile car il se réduit à un CVRP si chaque grappe comprend un seul nœud ou à un GmTSP si la capacité des véhicules est illimitée. Le nombre de papiers sur le GVRP est assez limité. Le problème a été introduit initialement par Ghiani et Laporte [24]. En 2003, Kara et Bektaş [36] ont proposé la première formulation de taille polynomiale. Un algorithme de colonies de fourmis et un algorithme génétique ont été proposés par Pop et al. [47] et Matei et al. [42] respectivement. Récemment, Bektaş et al. [9] ont proposé 4 formulations suivies par 4 algorithmes de branchement et coupes. Ils ont conclu que la meilleure formulation est une formulation non orientée à deux indices avec un nombre exponentiel

de contraintes. Une heuristique de type recherche locale à grand voisinage (LNS ou Large Neighbourhood Search) a été aussi proposée dans ce papier pour fournir des bornes supérieures pour les algorithmes de branchement et coupes. Dans la même période que Bektaş et al. [9], Pop et al. [48] ont introduit deux nouvelles formulations. La première est une formulation basée sur les nœuds et similaire à la formulation de Kara et Bektaş [36] mais produit une borne inférieure plus forte tandis que la seconde est une formulation basée sur les flots. Dans ce travail, une seule instance de Ghiani et Laporte [24] a été résolue directement par CPLEX. Aucune autre expérience de calcul avec les formulations proposées n'a été publiée et aucun algorithme de branchement et coupes n'a été développé.

Le GVRP a des applications principalement dans la planification des réseaux de distribution et dans la conception de réseaux de télécommunications. Quelques domaines d'applications immédiates sont énumérées ci-dessous (voir Baldacci et al. [6], Bektaş et al. [9] et Pop et al. [48]) :

- Routage des navires dans le transport maritime : étant donné un certain nombre de régions dans lesquelles sont localisés des ports, l'organisation de la distribution via ces ports peut être telle que les navires livrent les marchandises à un seul port dans chaque région (port principal), et les marchandises sont ensuite distribuées dans toute la région. Ce problème de routage peut être modélisé comme un GVRP où les régions (avec leurs ports respectifs) correspondent aux grappes et la flotte de navires correspond à l'ensemble de véhicules.
- Distribution de produits médicaux : une application spécifique du GVRP se pose quand un certain nombre de districts, qui englobent chacun un certain nombre de municipalités, ont besoin d'être approvisionnées en produits pharmaceutiques et qu'il suffit de fournir tout un lot de produits à une municipalité dans chaque district. Dans ce cas, le problème de la distribution correspond à un GVRP dans lequel les districts représentent les grappes et l'équipe de distribution médicale correspond à l'ensemble des véhicules.
- Problème de collecte des déchets urbains : ce problème consiste à concevoir des routes pour un certain nombre de véhicules qui sont utilisés pour ramasser les déchets urbains et les livrer à un centre d'enfouissement, un incinérateur ou une usine de recyclage à un coût de transport minimum.
- Conception de réseau de télécommunication sécurisé : étant donné une infrastructure de télécommunication centralisée avec un serveur central et des grappes d'ordinateurs. La conception doit être faite de telle sorte qu'exactement un ordinateur dans chaque grappe soit connecté avec le serveur central, et qu'il existe exactement deux chemins qui n'ont aucune arête commune entre le nœud central et chaque grappe. La sécurité du réseau est assurée par ces deux chemins où l'un doit être activé si l'autre est en panne. De toute évidence, toutes les solutions réalisables du GVRP peuvent être utilisées pour mettre en œuvre une telle conception, où le serveur central est le dépôt.

2.2 Problème de la tournée couvrante

Le problème de la tournée couvrante (CTP ou Covering Tour Problem) est équivalent au GTSP si les grappes dans le GTSP sont jointes et est défini comme suit :

Soit $G = (V \cup W, E_1 \cup E_2)$ un graphe non orienté, où $V \cup W$ est un ensemble de nœuds et $E_1 \cup E_2$ est un ensemble d'arêtes. $V = \{v_0, ..., v_{n-1}\}$ est l'ensemble des n nœuds qui peuvent être visités, et $W = \{w_1, w_2, ..., w_l\}$ est l'ensemble des l

nœuds qui doivent être couverts. Dénotons $T = \{v_0, ..., v_{t-1}\}$, un sous-ensemble de V, correspondant à l'ensemble des nœuds qui doivent être visités. v_0 est le dépôt. Un coût c_{ij} est associé à chaque arête de $E_1 = \{v_i, v_j : v_i, v_j \in V, i < j\}$ et une distance d_{ij} est associée à chaque arête de $E_2 = \{v_i, v_j : v_i \in V \setminus T, v_j \in W\}$. La mission du CTP est de déterminer une route fermée commençant et terminant au dépôt telle que :

1. chaque nœud de T soit visité exactement une fois et chaque nœud de $V \setminus T$ soit visité au plus une fois ;
2. chaque nœud de W soit couvert par la route, c'est-à-dire, se situe à une distance maximum r d'au moins un nœud de $V \setminus T$ qui est visité, où r est le rayon de couverture.

Si des contraintes de capacité sont considérées, nous avons la version à multiples véhicules (mCTP ou multi-vehicle Covering Tour Problem). Dans ce genre de problème, les deux contraintes suivantes sont normalement ajoutées en pratique :

1. le nombre de nœuds sur chaque route (exceptant le dépôt) doit être inférieur à une valeur donnée p ;
2. la longueur de chaque route ne dépasse pas une limite fixe q.

Si le GTSP a succité de nombreuses recherches, les travaux concernant le CTP sont beaucoup plus limités. Il semble que le premier travail sur ce problème ait été introduit par Current [17] en 1981. La version avec un seul véhicule (CTP) a ensuite été résolue exactement par un algorithme de branchement et coupes par Gendreau et al. [23]. Ils ont également proposé une heuristique. Baldacci et al. [5] ont utilisé une formulation de flux de deux marchandises (two-commodity flow formulation) et développé un algorithme de recherche par dispersion (ou scatter search algorithm). Pour la version avec multiples véhicules (mCTP), Hachicha et al. [30] ont introduit une formulation mathématique de flux avec trois indices et ont développé trois heuristiques inspirées par des algorithmes classiques : algorithme de Clarke et Wright [13], algorithme de balayage de Gillett et Miller [25] ("sweep algorithm"), et un algorithme de type partition d'abord - route ensuite Beasley [8] ("cluster-first/route-second"). Ces trois heuristiques ont été comparées entre elles, mais leur distance à l'optimalité est inconnue. Récemment, Jozefowiez [35] a proposé le premier algorithme exact pour le mCTP. Il est basé sur l'approche par génération de colonnes dans laquelle le problème maître est un problème de couverture d'ensemble ("set covering problem"), et le sous-problème est formulé de manière similaire au modèle du CTP proposé dans Gendreau et al. [23]. L'algorithme a été testé sur des instances issues de TSPLIB, et des résultats pour $|V|+|W|=100$ et $|T|=1$ ont été publiés.

Le mCTP permet de modéliser des problèmes de conception des réseaux de transport à deux niveaux où le tour correspond à une route primaire de véhicule, et tous les pointes qui ne se localisent pas sur cette route peuvent être facilement accessibles à partir des pointes visités par la route. Le problème de localisation des boîtes postales est un exemple d'application du mCTP (voir Labbé and Laporte [38]) : on doit localiser simultanément des boîtes aux lettres dans un ensemble V (de façon à ce que tous les habitants de la zone soient à une distance raisonnable d'une boîte) et construire des routes de collecte optimales. Un autre exemple est la conception des routes pour les équipes de soins mobiles dans les pays en voie de développement (voir Hodgson et al. [32], Swaddiwudhipong et al. [59]). Les services sont dispensés

par des équipes médicales dans un certain nombre de lieux choisis, et la population vivant en dehors de ces emplacements doit pouvoir se déplacer à pied pour les atteindre. Des problèmes similaires sont rencontrés dans plusieurs pays occidentaux par des équipes de prévention des maladies, dans l'industrie laitière Simms [55], et par les bibliothèques mobiles ou les systèmes bancaires.

Nous notons aussi qu'il y a un autre problème qui est considéré comme la version continue du CTP. Le problème est appelé "le voyageur de commerce suffisamment proche sur le plan"' (CETSP in the plane). Par rapport au CTP, l'ensemble V dans ce problème est maintenant continu. C'est-à-dire que les nœuds appartenant au tour peuvent se localiser n'importe où sur le plan. Des heuristiques ont été proposées par Gulczynski et al. [27] et Dong et al. [18]. Le problème a été étudié ensuite de manière plus détaillée dans la thèse de Mennell [43].

Le voyageur de commerce suffisamment proche sur le plan a une application pratique dans des opérations des véhicules aériens sans pilote, pour la reconnaissance aérienne et la distribution de munitions à des cibles. Par exemple, si un véhicule aérien sans pilote porte un capteur permettant de visualiser une zone surveillée sous forme de disque, le véhicule aérien doit se rapprocher, suffisamment de la cible pour qu'elle se trouve dans le disque visible du capteur. La cible peut alors être considérée comme visitée. Des applications similaires existent dans la reconnaissance sous-marine des côtes ou dans la lecture des compteurs intelligents.

2.3 Problème généralisé de tournées sur arcs

Tandis que dans les problèmes précédents, l'emplacement des tâches à effectuer se trouvent en des points précis sur un réseau (problème de tournées sur nœuds), les tâches du problème généralisé du postier rural sont situées sur des arcs ou arêtes. Il s'agit d'un problème de tournées sur arcs.

Le problème de tournées de véhicules sur arcs est un problème important de l'optimisation combinatoire. Les recherches sur ce problème ont été détaillées dans le livre Dror [20] en 2000 (pour les travaux publiés avant 2000) ou ont été résumées dans l'article de revue de la littérature de Corberán et Prins [14] (pour les travaux après 2000). Le problème du postier rural (RPP ou Rural Postman Problem) est le problème de base. Un postier doit commencer et terminer à une même localisation et chercher le plus court tour pour parcourir un sous-ensemble A_R de liens requis.

Le problème généralisé de la tournée sur arcs (GARP ou Generalized Arc Routing Problem) est une généralisation du RPP et est défini comme suit. Etant donné un graphe orienté $G = (V, A)$, avec l'ensemble de n nœuds $V = \{v_0, ..., v_{n-1}\}$, et l'ensemble de m arcs, $A = \{(v_i, v_j) : v_i, v_j \in V\}$. La numérotation des arcs nous permet d'exprimer A comme $\{a_1, a_2, ..., a_m\}$. Soit c_a le coût associé avec chaque arc de A. $C = \{C_0, ..., C_{K-1}\}$ est un ensemble de K grappes disjointes qui contiennent des arcs de A tels que chaque arc $a_i \in A$ appartient à exactement une grappe. C_0 comprend seulement le dépôt v_0. Le problème généralisé de la tournée sur les arcs consiste à déterminer un tour fermé de coût minimal traversant au moins un arc de toutes les grappes $C_k \in C$ au moins une fois.

Nous présentons ensuite un problème plus général appelé le problème généralisé de tournées sur arcs multi-véhicules (mGARP ou multi-vehicle Generalized Arc Routing Problem). Dans ce problème, des quantités (à livrer ou à collecter) sont associées aux grappes. Chaque véhicule a une capacité limitée et la somme des demandes traitées par une tournée ne dépasse pas cette capacité. Une autre contrainte

de capacité est aussi considérée dans la réalité pour limiter la longueur d'une tournée.

Jusqu'à maintenant, il n'existe que deux travaux dans la littérature qui traitent la version avec un seul véhicule du problème. Le premier a été présenté dans la thèse de Drexl [19] où le problème, défini sur un graphe orienté, est formulé et résolu par un algorithme de branchement et coupes et des heuristiques constructives. Le deuxième, qui a été présenté récemment par Aráoz et al. [1] dans le congrès ROUTE 2011, propose une formulation et quelques familles d'inégalités valides pour la version définie sur un graphe non orienté du problème.

2.4 Problème de la tournée sur arcs suffisamment proche

Nous discutons ici une généralisation du RPP que nous nommons le problème de la tournée sur arcs suffisamment proche (CEARP). Dans ce problème, l'ensemble des arcs requis A_R n'est pas défini explicitement. Il y a plutôt un ensemble de clients $W = \{w_1, w_2, ..., w_l\}$ qui doivent être couverts. Ces clients peuvent être localisés n'importe où dans la zone couverte par le réseau, et pas seulement sur le réseau. Comme dans la version originale du problème (voir Golden et al. [26]), nous considérons l'inclusion du dépôt, représenté par le nœud v_0. Le CEARP consiste à chercher un tour de coût minimum, commençant et terminant au dépôt, tel que chaque client de W soit couvert par le tour, c'est-à-dire, se trouve à une distance r d'un arc de la tournée.

La littérature existante pour le CEARP est assez limitée. Le problème a été introduit initialement par Golden et al. [26]. Les auteurs l'appellent *problème du voyageur de commerce sur le réseau routier* (CETSPsn ou Close-Enough Traveling Salesman Problem over a street network) et proposent quatre heuristiques pour résoudre 18 instances réelles avec une moyenne d'environ 900 segments routiers et 9000 clients pour chacune. Le rayon de couverture est testé avec deux valeurs : 100 et 150 m. Les heuristiques de Golden et al. [26] sont fondamentalement implémentées par un processus en 2 étapes. Dans une première étape, l'heuristique cherche un sous-ensemble d'arcs à traverser, soit par des procédures gloutonnes soit en résolvant un programme linéaire en nombres entiers. Dans la deuxième étape, le problème devient un RPP bien connu et est résolu par un solveur commercial.

Le CEARP est équivalent au problème généralisé de postier rural (GRPP) si les grappes dans le GRPP sont jointes. La version sans dépôt de ce type de GRPP est traitée dans Drexl [19], qui présente des méthodes de résolution (exacte et heuristiques). La méthode exacte est un algorithme de branchement et coupes basé sur une formulation mathématique tandis que les heuristiques sont des algorithmes d'étiquette dans lesquels la procédure de dominance est simplifiée de façon à ce que les heuristiques générées soient polynomiales et plus rapides.

Le CEARP peut être étendu en considérant que chaque véhicule a une capacité Q. Cette capacité peut être exprimée par la longueur maximale de route que les véhicules peuvent couvrir ou la durée de travail des techniciens. La capacité limitée des véhicules oblige à construire plusieurs tournées. Ce problème est nouveau, avec des applications pratiques et dénommé "problème de la tournée sur arcs suffisamment proche avec multiples véhicules" (mCEARP ou multiple-vehicle Close Enough Traveling Salesman Problem).

L'application principale de ce problème est de construire des routes pour le relevé des compteurs (voir Golden et al. [26] pour plus d'information). Afin de me-

surer l'information de consommation des clients à distance, un appareil de lecture est installé sur un véhicule pour collecter des données envoyées par les compteurs intelligents. Le technicien ne doit pas visiter chaque client pour lire les compteurs mais juste entrer dans la zone de couverture des compteurs. Le véhicule doit traverser la zone de service et passer suffisamment proche de chaque compteur de façon à ce qu'ils soient tous lus. Le rayon de lecture r est normalement entre 150 et 300 m mais peut atteindre 381 m (voir Golden et al. [26]).

2.5 Conclusion

Nous avons introduit dans les paragraphes précédents une vue panoramique des problèmes généralisés de tournées de véhicules. Nous avons défini les problèmes, résumé les travaux de la littérature dans ces domaines et présenté leurs applications réelles.

Cette introduction des problèmes généralisés de tournées de véhicules nous a conduit à quelques conclusions :

- Tous les problèmes généralisés sont NP-difficiles. Ils sont plus difficiles que les problèmes de tournée de véhicules classiques car leurs modèles sont plus généraux et contiennent les autres problèmes comme cas particuliers.
- Malgré le rôle de plus en plus important des problèmes généralisés en applications réelles, la littérature pour ce genre des problèmes est assez restreinte. Plusieurs problèmes n'ont jamais été traité, par exemple : GmTSP, mGARP et mCEARP.
- Il est possible d'améliorer les méthodes et les algorithmes proposés en terme de qualité et de temps de calcul. De nouvelles méthodes exactes peuvent être proposées par le développement de l'algorithme de branchement et coupes basé sur de nouvelles formulations mathématiques. De nouvelles métaheuristiques peuvent aussi être proposées pour fournir de bonnes solutions pour des instances réelles dans des temps raisonnables.

L'ensemble de ces conclusions a conduit aux recherches qui sont exposées dans les chapitres suivants. Notre motivation est de traiter des problèmes les plus généraux possibles ou des problèmes peu ou pas traités par la communauté scientifique. Ainsi cette thèse traitera de trois problèmes généralisés de tournées de véhicules : mCTP, GVRP et CEARP, présentés successivement dans les trois chapitres suivants.

3

Problème de la tournée sur arcs suffisamment proche

Dans ce chapitre, nous introduisons la résolution pour le problème de la tournée sur arcs suffisamment proche (CEARP). Nos contributions sont les suivantes : 1) nous proposons deux nouvelles formulations pour le CEARP ; 2) nous les comparons, à la fois analytiquement et empiriquement avec une autre formulation présentée dans Drexl [19] ; 3) nous améliorons l'algorithme de branchement et coupes de Drexl [19] et proposons une heuristique pour le CEARP ; et 4) nous proposons aussi un algorithm constructif basé sur la PLNE pour résoudre des instances réelles.

Avant de proposer des formulations ainsi que des méthodes de résolution pour le CEARP, nous redéfinissons le problème et présentons les notions utilisées dans ce chapitre.

Dénotons $G = (V, A)$ un graphe orienté, où $V = \{v_0, ..., v_{n-1}\}$ est un ensemble de nœuds, le nœud v_0 est un dépôt, et $A = \{(v_i, v_j) : v_i, v_j \in V\}$ est une famille de couples ordonnés de nœuds appelés arcs. Si nous numérotons les arcs, A peut être exprimé comme $A = \{a_1, a_2, ..., a_m\}$. Un coût c_a est associé à chaque arc a (distance ou temps de parcours). Définissons $W = \{w_1, w_2, ..., w_l\}$, un ensemble de clients qui doivent être couverts. Le but du CEARP est de chercher un tour le plus court commençant et se terminant au dépôt tel que chaque client soit couvert par le tour, c'est-à-dire la distance minimale entre chaque client et le tour est inférieure ou égale à r, le rayon de lecture.

Etant donné un sous-ensemble de nœuds, $S \subseteq V$, dénotons $\delta^+(S)$ un ensemble d'arcs sortant de S et $\delta^-(S)$ un ensemble d'arcs entrant dans S. Si $S = \{v_k\}$, nous écrivons simplement $\delta^+(k)$ (ou $\delta^-(k)$) au lieu de $\delta^+(\{v_k\})$ (ou $\delta^-(\{v_k\})$). $A(S)$ est l'ensemble d'arcs avec les deux extrémités dans S. Nous définissons les coefficients binaires λ_{lk} égale à 1 si et seulement si le client $w_l \in W$ peut être couvert par l'arc $a_k \in A$. Etant donné $x \in N^{|A|}$ et $T \subset A$, $x(T)$ dénote $\sum_{e \in T} x_e$.

Le chapitre est organisé comme suit : nous commençons par proposer des formulations pour le problème. Des méthodes de résolution, des évaluations numériques et des conclusions sont ensuite présentés. Enfin nous présentons nos deux articles concernés qui donnent tous les détails des méthodes et des résultats.

3.1 Des formulations mathématiques pour le CEARP

Nous introduisons d'abord une nouvelle formulation dénotée F1 pour le CEARP. Cette formulation a fait l'objet d'une présentation en conférence internationale, *1st International Conference on Operations Research and Enterprise Systems*-ICORES 2012 (voir l'article 1 à la fin de ce chapitre). Dans cette formulation, les variables x_a représentent le nombre de fois que l'arc a est traversé.

$$\text{F1 :} \qquad \text{Minimiser} \sum_{a \in A} c_a x_a \tag{3.1}$$

$$\text{sous les contraintes :} \qquad x\Big(\delta^+(0)\Big) \geq 1 \tag{3.2}$$

$$x\Big(\delta^+(i)\Big) - x\Big(\delta^-(i)\Big) = 0 \qquad \forall i \in V \tag{3.3}$$

$$\sum_{a \in A} \lambda_{wa} x_a \geq 1 \qquad \forall w \in W \tag{3.4}$$

$$Mx\Big(\delta^+(S)\Big) - x\Big(A(S)\Big) \geq 0 \; \forall S \subset V - \{v_0\}$$
$$\text{et } 2 \leq |S| \leq n-2 \tag{3.5}$$

$$x_a \in Z^+ \qquad \forall a \in A \tag{3.6}$$

où M est un très grand nombre représentant une borne supérieure sur le nombre de fois que les arcs de S sont utilisés.

L'objective (3.1) est de minimiser le coût de parcours total. La contrainte (3.2) implique que le dépôt appartient au tour, les contraintes (3.3) assurent la conservation de flux. Les contraintes (3.4) assurent que tous les clients de W sont couverts par le tour, et les contraintes (3.5) sont celles de connectivité. Ces contraintes assurent la présence d'au moins un arc sortant de n'importe quel ensemble S, pour tous les sous-ensembles possibles S de V contenant un arc appartenant au tour. Les contraintes (3.6) définissent les variables.

Les contraintes de connectivité peuvent être écrites sous une autre forme :

$$Mx\Big(\delta^+(S)\Big) - x_a \geq 0 \qquad \forall S \subset V - \{v_0\}$$
$$\text{et } 2 \leq |S| \leq n-2, \; a \in A(S) \tag{3.7}$$

où M est un très grand nombre représentant une borne supérieure sur le nombre de fois qu'un arc est utilisé. Comme dans Drexl [19], M peut être estimé par $|A| + 1$.

Bien que cette forme comporte plus de contraintes que la version précédente, nous pouvons borner le nombre M plus strictement par $|A|+1$ au lieu de $m(|A|+1)$. Des tests ont montré que l'utilisation des contraintes (3.7) était plus efficace et dans le cadre de cette thèse nous présentons des résultats pour cette forme de contraintes. Les résultats avec les contraintes (3.5) peuvent être trouvés dans l'article 1 à la fin de ce chapitre.

Comme le CEARP est équivalent au GARP, la formulation proposée dans Drexl [19] pour le GARP peut être appliquée directement au CEARP. Parce que la formulation dans Drexl [19] traite la version sans dépôt du problème, nous la modifions légèrement. Dans cette formulation, y_i est la variable binaire qui indique l'utilisation du nœud i dans la solution, et la variable entière x_a dénote le nombre de fois

que l'arc a est traversé. Soit he_a la tête du l'arc a. Une formulation pour le CEARP (F2) peut être décrite comme suit :

$$\text{F2 :} \quad \text{Minimiser} \quad \sum_{a \in A} c_a x_a \tag{3.8}$$

$$\text{sous les contraintes :} \quad y_0 = 1 \tag{3.9}$$

$$x\Big(\delta^+(i)\Big) - x\Big(\delta^-(i)\Big) = 0 \qquad \forall i \in V \tag{3.10}$$

$$\sum_{a \in A} \lambda_{wa} x_a \geq 1 \qquad \forall w \in W \tag{3.11}$$

$$x\Big(\delta^+(S)\Big) - y_i \geq 0 \qquad \forall S \subset V - \{v_0\},$$
$$\text{et } 2 \leq |S| \leq n-2,\ i \in S \tag{3.12}$$

$$x_a - M y_i \leq 0 \ \forall i \in V, a \in A \text{ avec } he_a = i \tag{3.13}$$

$$x_a \in Z^+ \text{et } y_i \in \{0,1\} \ \forall a \in A \text{ et } \forall i \in V \tag{3.14}$$

Dans cette formulation, les contraintes (3.9)–(3.12) ont la même signification que dans la formulation F1, et les contraintes (3.13) sont utilisées pour exprimer la relation entre les deux types de variables. Comme dans Drexl [19], M peut être fixé à $|A| + 1$.

Nous décrivons maintenant une troisième formulation pour le CEARP, que nous appelons F3. Soit y_a une variable binaire qui représente l'utilisation de l'arc a avec le service. Les variables entières x_a désignent le nombre de fois que l'arc a est utilisé sans service. Ensuite, le CEARP peut être formulé comme suit :

$$\text{F3 :} \quad \text{Minimiser} \quad \sum_{a \in A} c_a (x_a + y_a) \tag{3.15}$$

$$\text{sous les contraintes :} \quad x\Big(\delta^+(0)\Big) + y\Big(\delta^+(0)\Big) \geq 1 \tag{3.16}$$

$$x\Big(\delta^+(i)\Big) + y\Big(\delta^+(i)\Big) - x\Big(\delta^-(i)\Big) - y\Big(\delta^-(i)\Big) = 0 \qquad \forall i \in V \tag{3.17}$$

$$\sum_{a \in A} \lambda_{wa} y_a \geq 1 \qquad \forall w \in W \tag{3.18}$$

$$x\Big(\delta^+(S)\Big) + y\Big(\delta^+(S)\Big) - y_a \geq 0 \qquad \forall S \subset V - \{v_0\},$$
$$\text{et } 2 \leq |S| \leq n-2,\ a \in A(S) \tag{3.19}$$

$$x_a \in Z^+ \text{et } y_a \in \{0,1\} \qquad \forall a \in A. \tag{3.20}$$

La signification de chaque contrainte dans F3 est la même que dans F1. Les contraintes (3.16)–(3.19) impliquent les contraintes (3.2)–(3.7) respectivement.

Il est facile de montrer que les trois formulations F1, F2 et F3 sont équivalentes. Le tableau 3.1 donne une comparaison de ces trois formulations. F1 a le plus petit nombre de variables et de contraintes (nous ne considérons pas les contraintes de connectivité ici en raison de leur nombre exponentiel). L'inconvénient le plus important de F1 est qu'il utilise un grand nombre M dans la contrainte de connectivité. Parce que nous ne pouvons pas trouver un moyen efficace d'estimer M de

	F1	F2	F3
No. variables	$\|A\|$	$\|A\| + \|V\|$	$2.\|A\|$
No. contraintes	$\|V\| + \|W\| + 1$	$\|V\| + \|A\| + \|W\| + 1$	$\|V\| + \|W\| + 1$
Grand-M	Oui	Oui	Non

Tableau 3.1 – Comparaison des trois formulations

façon serrée, une procédure exacte de séparation de cette contrainte est inutile. Des expériences montrent que la performance de l'algorithme de séparation et coupes basé sur cette formulation est faible par rapport à celle des deux autres formulations. F3 a le plus grand nombre de variables, mais il a aussi quelques avantages remarquables. Tout d'abord, il ne contient pas les contraintes de grand M qui sont bien connues pour affaiblir la relaxation linéaire et pour diminuer la performance des modèles de programmation linéaire en nombre entier (PLNE). Deuxièmement, le polytope de la couverture d'ensemble (ou "set covering polytope" en anglais) avec des variables binaires a reçu plus d'attention que celui avec des variables entières. Par conséquent, l'identification des contraintes violées de type (3.18) dans F3 semble être plus favorable que celui de (3.4) dans F1 ou (3.11) dans F2. La formulation F2 a moins de variables, mais plus de contraintes que F3. En outre, elle contient des contraintes avec un grand M.

3.2 Méthodes de résolution proposées

A partir des formulations mathématiques proposées dans la section précédente, nous construisons des méthodes de résolution pour le CEARP. Dans cette section, nous ne les introduisons que brièvement. La description détaillée des méthodes peut être trouvée dans les articles 1 et 2 à la fin de ce chapitre.

Nous proposons d'abord de résoudre le CEARP à l'optimalité par un algorithme de plans sécants. Notre algorithme est basé sur la formulation F1. La stratégie principale est de résoudre de manière itérative une PLNE incluant les contraintes (3.2), (3.3), (3.4), et (3.6). A chaque itération, nous cherchons des "sous-tours" n'incluant pas le dépôt du graphe créés par la solution optimale actuelle. S'il y a des sous-tours trouvés, les contraintes de connectivité (3.7) violées sont ajoutées au modèle. Le modèle est résolu jusqu'à ce qu'il n'existe plus des sous-tours.

Nous résolvons aussi le problème de manière optimale en construisant trois algorithmes de branchement et coupes basés sur les trois formulations correspondantes. Afin de simplifier la description, nous décrivons l'algorithme seulement pour F3 ; l'implémentation est similaire pour F1 et F2. Nous résolvons principalement un programme linéaire contenant les contraintes (3.16), (3.17), (3.18) et les contraintes $0 \leq y_a \leq 1$. Ensuite, nous cherchons des contraintes qui violent des inégalités valides pour le CEARP. Les contraintes détectées sont ajoutées au programme actuel, qui est ré-optimisé. Ce processus est répété jusqu'à ce que toutes les contraintes soient satisfaites. S'il y a des variables fractionnaires, nous branchons pour générer deux nouveaux sous-problèmes. Si toutes les variables sont entières, nous exploitons un autre sous-problème. Des inégalités valides pour le CEARP, la façon de les séparer ainsi que l'implémentation des algorithmes de branchement et coupes sont décrites en détail dans l'article 2 à la fin de ce chapitre.

Nous utilisons une heuristique rapide appelée UB1 qui donne des solutions réalisables pour le CEARP. Cette heuristique est utilisée dans les algorithmes de

branchement et coupes pour fournir une borne supérieure initiale. Nous résolvons d'abord une programmation linéaire en nombres entiers (PLNE) comportant les contraintes (3.1), (3.3), (3.4) et (3.6) en utilisant CPLEX. La résolution dans nos tests de telle PLNE est rapide même pour de grandes instances dans nos tests. Et puis, nous construisons un graphe orienté $G_R = (V_R, A_R)$ induit par la solution de l'étape 1, ajoutant le dépôt s'il n'est pas présent encore dans le graphe. Si G_R est connecté alors arrêter ; la solution trouvée à l'étape 1 est aussi une solution réalisable pour le CEARP. Sinon, nous relions les composantes isolées dans G_R en utilisant l'algorithme proposé dans Ball and Magazine [7] pour résoudre le DRPP (ou Directed Rural Postman Problem).

Nous présentons ensuite le dernier algorithme appelé UB2 qui donne de bonnes solutions pour le CEARP. L'idée principale est à l'origine de l'observation suivante. Dans la pratique, les arcs sont habituellement parcourus seulement quelques fois, et le nombre de traversées est beaucoup plus faible que la valeur la plus petite prouvable de M. Cela donne à penser que nous pouvons borner le grand nombre M dans les algorithmes de branchement et coupes par une petite valeur pour améliorer la performance. Afin de s'assurer qu'il existe toujours une solution réalisable, un problème du postier chinois (CPP ou Chinese Postman Problem) est d'abord résolu, et M est déterminé par le nombre maximal de fois qu'un arc est traversé dans la solution du CPP. Nous n'utilisons que F2 et F3 pour développer cet algorithme car ils donnent des résultats meilleurs que F1.

3.3 Evaluation numérique

3.3.1 Implémentation et jeux de donnés

Nos algorithmes sont programmés en C/C++ et exécutés sur un CPU Intel Xeron 2.4 GHz. Les algorithmes de branchement et coupes et de plan sécant sont construits autour de CPLEX 11.2 en utilisant Callable Library. Le temps de calcul des algorithmes exacts (de branchement et coupes et de plans sécants) et l'algorithme UB2 est limité à deux heures pour chaque instance.

Nous testons les algorithmes sur les instances créées à partir de trois types de graphes : orientés, non orientés et mixtes. Afin de construire les instances orientées pour le CEARP, nous générons aléatoirement des graphes qui imitent les réseaux routiers réels par la procédure suivante :

- Les coordonnées de n nœuds sont générées aléatoirement dans un carré unitaire. Une heuristique est ensuite utilisée pour chercher un tour le plus court passant par tous les nœuds exactement une fois. Ce tour est un cycle Hamiltonien et utilisé comme un cadre pour construire le graphe complet. Pour cela, le graphe créé est fortement connexe.
- Afin d'imiter les réseaux réels, des arcs aléatoires sont ajoutés au tour actuel pour atteindre le nombre total d'arcs souhaités $m = nd$, où n représente le nombre de nœuds et d le ratio entre le nombre d'arcs et le nombre de nœuds, de telle sorte que : 1) les arcs ne sont pas trop longs, et 2) il n'y a pas d'intersection entre chaque paire d'arcs comme un réseau réel.

Dans nos tests, nous utilisons les graphes avec le nombre de nœuds $n \in \{300, 400, 500\}$ et le ratio entre le nombre d'arcs et le nombre de nœuds $d \in \{1.5, 2, 2.5, 3\}$. Pour chaque couple de n et d, nous générons 5 graphes différents. Les coûts des arcs sont définis comme c_{ij} kilomètres, où c_{ij} est la distance euclidienne entre v_i et

v_j multipliée par 5 pour obtenir la longueur moyenne des arcs proche de la réalité (d'environ 0,2 à 0,4 kilomètre)

Une fois que les graphes sont créés, les instances du CEARP sont générées par la localisation de $q = mt$ nœuds de clients sur le carré contenant le graphe, où m dénote le nombre d'arcs et t le ratio du nombre de clients sur le nombre d'arcs $t \in \{0, 5, 1, 5, 10\}$. De cette manière, pour chaque graphe, quatre instances du CEARP sont créées. Le rayon de couverture r est d'abord fixé à la valeur de 150 mètres. Afin d'assurer l'existence d'une solution réalisable, nous supprimons tous les clients qui ne peuvent être couverts par aucun arc. Nous examinons aussi l'impact de l'augmentation du paramètre r de 150 mètres à 200 mètres. Autrement dit, les arcs et les coordonnées des clients et des nœuds sont gardés constants tandis que r est augmenté.

Pour chaque valeur de r, nous générons 240 instances du CEARP nommées par ce-n-k-t, où n est le nombre de nœuds, k indique le nombre d'arcs et t le rapport entre le nombre de clients et le nombre d'arcs. Par exemple, ce-300-450-10 signifie une instance avec 300 nœuds, 450 arcs, et $t = 10$.

Nous utilisons aussi des graphes mixtes de la littérature pour générer des instances pour le CEARP. Nous choisissons les graphes mixtes introduits dans Corberán et al. [16] car les coordonnées de leurs nœuds sont publiées. Ce sont des graphes de grande taille avec une structure similaire à celle des réseaux routiers réels. Pour transformer les graphes mixtes en graphes orientés, nous modélisons chaque arête non orientée par deux arcs avec le même coût. A partir de ces graphes, nous sélectionnons deux graphes, MB537 et MB547, qui, après avoir été transformés en graphes orientés, ont moins de 1500 arcs. MB357 a 500 nœuds, 364 arêtes et 476 arcs et MB547 a 500 nœuds, 351 arêtes et 681 arcs.

Pour les graphes mixtes, la procédure générant des clients est identique, sauf que le rayon de couverture r est maintenant déterminé par la longueur moyenne de tous les arcs dans le graphe. Pour chaque graphe et chaque valeur de t, nous générons aussi cinq instances du CEARP.

Finalement, nous testons nos algorithmes sur 30 instances définies sur des graphes non orientés. Nous utilisons deux ensembles de graphes proposés dans Corberán et al. [15] avec 15 instances du problème de tournée général (GRP ou General Routing Problem) pour chacun, générées à partir de deux graphes d'Albaida et de Madrigueras introduits dans Benavent et al. [11] en définissant chaque arête comme étant requise avec une probabilité de $P \in \{0,3; 0,5; 0,7\}$. Le graphe d'Albaida comporte 116 nœuds et 174 arêtes, et le graphe de Madrigueras a 196 nœuds et 316 arêtes. A partir de ces graphes, nous générons les instances du CEARP comme suit :

- Le nombre de clients est défini par le nombre d'arêtes obligatoires dans le graphe ; chaque client est couvert par une arête obligatoire.
- Chaque client est couvert par e arêtes obligatoires supplémentaires, où e est une valeur aléatoire de 1 à 5, telle que chaque client soit couvert par au moins 2 et au plus 6 arêtes.

Afin de résoudre les instances non orientées comme les instances orientées, nous transformons chaque arête en deux arcs avec le même coût. Comme dans les autres problèmes de tournées de véhicules définis sur les graphes non orientés (voir Ghiani et Laporte [24], par exemple), il est facile de montrer que, pour une instance donnée du CEARP définie sur un graphe non orienté, une solution optimale existe dans laquelle aucune arête n'est traversée plus de deux fois. Cela nous permet de fixer le coefficient M à 2 dans F2 et à 1 dans F3.

3.3.2 Pourquoi nos instances sont-elles difficiles ?

Puisque le CEARP se réduit au RPP qui peut encore se réduire à un problème facile de postier chinois orienté (DCPP ou Directed Chinese Postman Problem), il est possible que nos instances orientées soient proches du DCPP et donc faciles à résoudre si nous générons trop de clients et si tous les arcs du graphe doivent être activés pour assurer la couverture. Pour clarifier cela, nous utilisons une procédure d'élimination des clients pour voir le nombre de clients restants. Cette procédure est utilisée dans tous les algorithmes car elle nous permet aussi de réduire la taille du problème (au moins de 50% des clients dans nos tests).

Etant donné $w_l \in W$, soit $Z(w_l)$ l'ensemble des arcs qui peuvent couvrir w_l. Considérons chaque paire de clients w_i et w_j ; si $Z(w_i) \subseteq Z(w_j)$, le client w_j peut alors être éliminé. La raison est que si l'on peut servir w_i, w_j est couvert en même temps. Notons que, le nombre de clients restants après la réduction est aussi le nombre maximum d'arcs qui doivent être activés pour assurer la couverture. Les figures 3.1 et 3.2 illustrent une instance avec 500 nœuds et 750 arcs avant et après la réduction. Ici, le graphe est représenté par les lignes pleines et les clients sont les points.

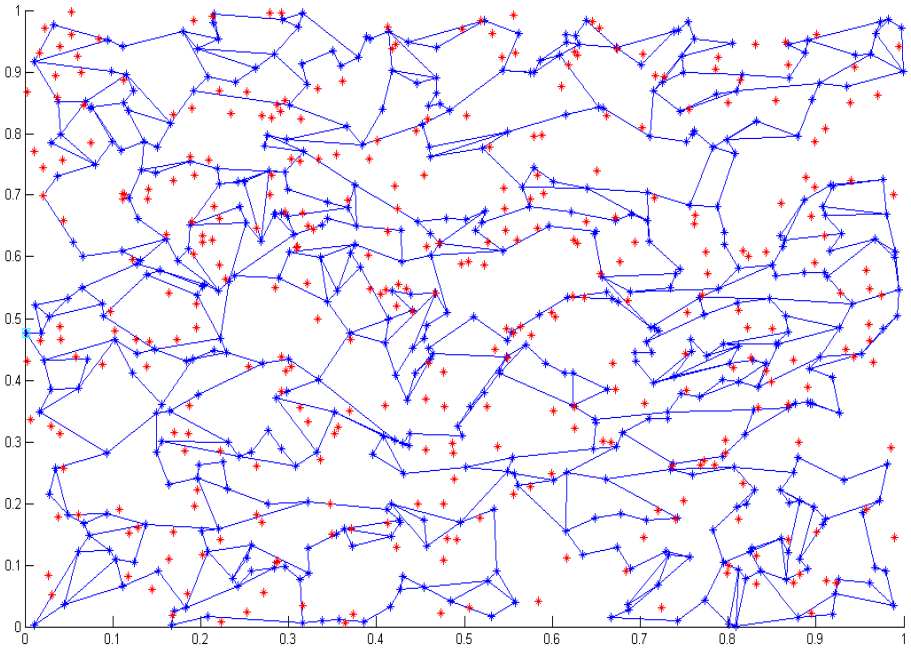

FIGURE 3.1 – Une instance avec 500 nœuds et 750 arcs avant la réduction

La complexité de nos instances est représentée par les chiffres du tableau 3.2. Ici, la colonne "Instance" indique le nom d'instance et la colonne "Client Av." présente le nombre moyen de clients avant la procédure de réduction. Ensuite, les colonnes "Client Ap.", "Arc Obli." et "Arc Moy." représentent le nombre moyen de clients après la réduction, le nombre moyen d'arcs obligatoires (un arc est dit obligatoire s'il couvre un client unique qui ne peut être couvert par aucun de tous les autres arcs) et le nombre moyen d'arcs couvrant un client respectivement pour deux valeurs de r.

		r=150			r=200		
Instance	Client Av.	Client Ap.	Arc Obli.	Arc Moy.	Client Av.	Arc Obli.	Arc Moy.
ce300-450-0.5	171.4	104.8	27.8	3.2	102.6	7.0	4.4
ce300-450-1	344.0	141.2	46.0	2.8	141.0	13.0	4.1
ce300-450-5	1737.8	203.2	110.4	1.9	194.4	33.8	3.3
ce300-450-10	3466.6	211.0	141.6	1.6	204.0	45.6	3.1
ce300-600-0.5	245.2	125.4	31.8	3.8	121.6	8.8	5.9
ce300-600-1	490.2	167.6	53.8	3.4	171.4	15.0	5.5
ce300-600-5	2436.4	222.2	116.8	2.2	217.0	37.6	4.4
ce300-600-10	4855.6	234.0	145.2	1.9	221.4	47.6	4.0
ce300-750-0.5	313.8	142.0	33.2	4.4	153.8	10.0	7.3
ce300-750-1	640.2	187.6	54.8	3.9	197.0	17.2	6.7
ce300-750-5	3223.0	251.4	113.2	2.7	243.0	38.0	5.5
ce300-750-10	6445.8	263.2	136.4	2.4	246.2	47.8	5.0
ce300-900-0.5	402.6	169.6	36.6	5.2	172.4	12.4	8.3
ce300-900-1	809.4	213.6	56.4	4.6	212.6	18.8	7.8
ce300-900-5	4054.4	278.4	103.4	3.3	259.6	38.2	6.5
ce300-900-10	8120.4	294.8	117.0	3.1	269.2	47.2	6.0
ce400-600-0.5	249.6	142.2	25.2	3.5	145.6	6.8	5.3
ce400-600-1	502.0	192.4	44.6	3.3	205.4	11.4	5.2
ce400-600-5	2512.8	260.0	116.6	2.3	273.8	29.8	4.4
ce400-600-10	5017.0	270.8	149.6	2.0	272.8	37.0	4.0
ce400-800-0.5	349.2	175.8	30.0	4.5	182.0	8.2	7.3
ce400-800-1	703.4	221.6	51.0	4.1	237.6	13.2	6.9
ce400-800-5	3494.0	295.2	115.8	3.1	320.4	28.4	6.0
ce400-800-10	6992.2	307.2	143.0	2.7	325.6	34.8	5.6
ce400-1000-0.5	457.6	198.0	31.6	5.4	208.0	8.0	8.7
ce400-1000-1	918.6	253.6	50.2	4.8	270.8	12.6	8.3
ce400-1000-5	4561.8	316.6	110.6	3.7	333.4	27.8	7.3
ce400-1000-10	9131.8	328.8	132.0	3.3	338.2	34.6	6.9
ce400-1200-0.5	564.4	224.8	33.0	6.3	239.0	9.6	10.6
ce400-1200-1	1131.0	280.4	51.8	5.7	295.0	14.2	10.0
ce400-1200-5	5633.2	350.4	96.8	4.6	383.2	30.2	9.0
ce400-1200-10	11260.4	365.8	114.2	4.3	395.0	37.4	8.7
ce500-750-0.5	330.8	184.2	25.8	4.0	191.2	5.0	6.2
ce500-750-1	664.4	246.8	44.4	3.7	264.8	9.6	6.1
ce500-750-5	3344.2	319.2	107.6	2.7	369.6	20.8	5.4
ce500-750-10	6682.2	329.2	141.4	2.5	386.0	26.6	5.2
ce500-1000-0.5	461.0	223.4	31.4	5.3	235.8	5.2	8.4
ce500-1000-1	920.8	285.2	52.0	4.8	310.8	10.4	8.0
ce500-1000-5	4613.6	358.0	120.4	3.7	412.0	23.6	7.2
ce500-1000-10	9222.4	375.6	146.4	3.4	424.8	30.6	6.9
ce500-1250-0.5	470.8	260.6	30.0	6.7	282.6	6.6	10.6
ce500-1250-1	1182.0	329.2	48.8	6.1	363.0	10.0	10.3
ce500-1250-5	5893.4	396.8	104.6	4.9	484.2	21.4	9.4
ce500-1250-10	11807.2	415.6	124.8	4.5	500.0	26.8	9.1
ce500-1500-0.5	718.4	305.0	26.0	8.0	313.6	9.6	11.9
ce500-1500-1	1433.6	354.2	45.0	7.3	387.2	15.6	11.3
ce500-1500-5	7157.0	450.8	94.2	6.2	513.4	33.2	10.4
ce500-1500-10	14327.0	469.6	109.8	5.9	532.8	37.8	10.2

Tableau 3.2 – Caractéristiques d'instances générées sur les graphes orientés

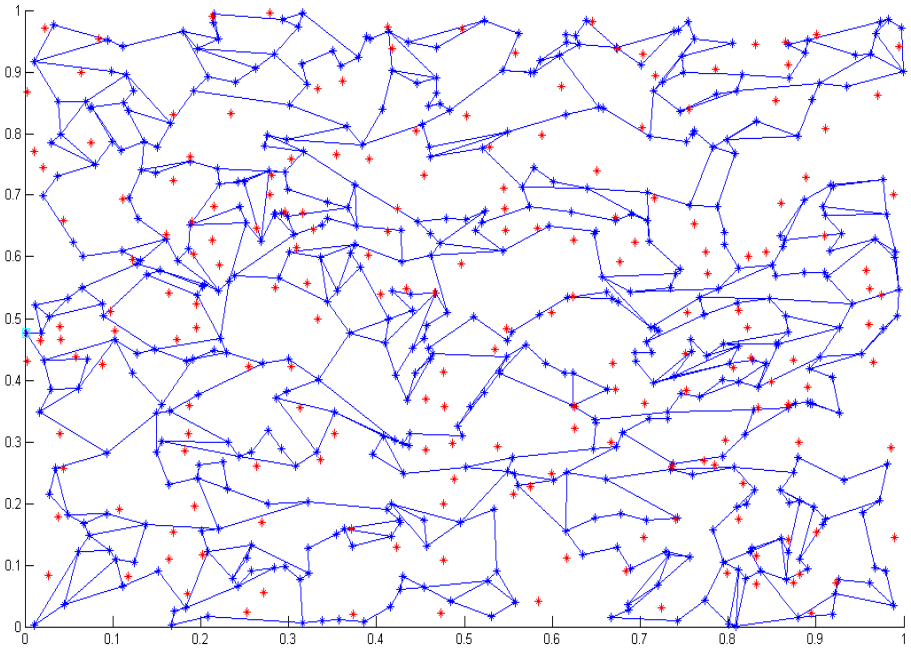

FIGURE 3.2 – Une instance avec 500 nœuds et 750 arcs après la réduction

Les résultats du tableau 3.2 indiquent que le ratio entre le nombre de clients restants et le nombre d'arcs dans le graphe est normalement de 0,2 à 0,5. Par conséquent, le nombre d'arcs qui doivent être activés pour l'objectif de couverture est plus petit que le nombre total d'arcs. En outre, le ratio maximal entre le nombre d'arcs obligatoires et le nombre total d'arcs dans le graphe est de 0,3. C'est pour ces raisons que nos instances sont loin d'un DCPP, i.e. difficiles.

Une autre observation intéressante est que si le ratio t diminue ou si le rayon de couverture r augmente, le nombre d'arcs obligatoires diminue et le nombre d'arcs moyen couvrant un client monte. Cela peut conduire à une prévision que la difficulté du problème augmente proportionnellement à r et inversement proportionnellement à t.

3.3.3 Résultats des évaluations numériques

Comparaison de deux méthodes basées sur F1

Dans ce paragraphe, nous comparons deux méthodes exactes : plan sécant (CP-F1) et branchement et coupes (B&C-F1) qui sont construites sur F1. L'expérimentation est effectuée sur les instances orientées. Le temps de calcul des deux algorithmes est limité à 2 heures.

Les tableaux 3.3 et 3.4 présentent les résultats de comparaison de deux méthodes exactes basées sur F1 pour deux valeurs différentes du rayon de couverture r. Ils comportent le nom des instances dans la première colonne "Instance". La colonne "Succ" indique le nombre de solutions optimales obtenues pour chaque type d'instances. Ensuite, les colonnes "Itér" et "Nœud" présentent respectivement le nombre moyen d'itérations de PLNE dans l'algorithme de plant sécant et le nombre

	CP-F1			B&C-F1		
Instance	Succ	Itér	Temps	Succ	Nœud	Temps
ce300-450-0.5	5	4.8	0.10	5	3.6	0.07
ce300-450-1	5	3.6	0.13	5	1	0.12
ce300-450-5	5	1.4	0.95	5	0.4	0.91
ce300-450-10	5	1.4	2.13	5	0.2	2.02
ce300-600-0.5	5	16	1.17	5	184	0.44
ce300-600-1	5	9.2	0.64	5	111	0.49
ce300-600-5	5	7.2	2.48	5	16	2.31
ce300-600-10	5	1.8	5.32	5	5.4	5.05
ce300-750-0.5	5	49	130.60	5	74343.4	1426.93
ce300-750-1	5	6.6	1.45	5	4567.8	6.49
ce300-750-5	5	3.6	4.97	5	103.8	4.75
ce300-750-10	5	2.8	10.91	5	6.8	10.43
ce300-900-0.5	5	80.80	1285.14	2	205409.5	614.99
ce300-900-1	5	12	40.57	3	13107.7	28.69
ce300-900-5	5	4.0	9.76	5	2183	17.71
ce300-900-10	5	3.2	20.06	5	323	20.01
ce400-600-0.5	5	19.6	0.45	5	5.2	0.21
ce400-600-1	5	3.2	0.35	5	6.0	0.37
ce400-600-5	5	1.4	2.79	5	0.2	2.40
ce400-600-10	5	1.2	6.41	5	0.0	6.39
ce400-800-0.5	5	56.6	69.20	5	13050.4	162.64
ce400-800-1	5	7.6	1.23	5	183.4	1.34
ce400-800-5	5	2.6	6.80	5	18.4	6.86
ce400-800-10	5	2.2	15.71	5	12.2	15.45
ce400-1000-0.5	3	12.0	143.25	1	7557	18.79
ce400-1000-1	5	93.2	504.72	4	37923.3	153.56
ce400-1000-5	5	4.0	14.54	5	1092.2	14.77
ce400-1000-10	5	3.4	29.79	5	385.8	28.83
ce400-1200-0.5	0			0		
ce400-1200-1	4	8.25	288.95	2	300073.0	2430.52
ce400-1200-5	5	5.2	32.31	5	42259.4	215.38
ce400-1200-10	5	5.6	59.28	5	4484.6	60.79
ce500-750-0.5	5	3.8	3.32	5	37.6	0.53
ce500-750-1	5	3.4	0.83	5	8.0	0.88
ce500-750-5	5	2.2	6.57	5	4.0	6.61
ce500-750-10	5	2.0	14.70	5	1.8	14.79
ce500-1000-0.5	5	10.83	45.78	4	68889.8	574.95
ce500-1000-1	5	7.2	6.17	5	3.09	12.47
ce500-1000-5	5	5.0	16.17	5	491.0	16.81
ce500-1000-10	5	4.6	34.86	5	76.2	34.84
ce500-1250-0.5	0			0		
ce500-1250-1	3	13.67	2277.41	1	56230	503.48
ce500-1250-5	5	11.6	49.94	5	41311.2	216.61
ce500-1250-10	5	6.6	73.67	5	18255.0	127.39
ce500-1500-0.5	0			0		
ce500-1500-1	1	6.0	6646.2	0		
ce500-1500-5	4	8.3	24.19	3	239353.7	2597.60
ce500-1500-10	5	5.6	155.67	5	227611.8	2032.38

Tableau 3.3 – Comparaison de deux méthodes basées sur F1 sur les instances avec $r = 150\,\mathrm{m}$

	CP-F1			B&C-F1		
Instance	Succ	Itér	Temps	Succ	Nœud	Temps
ce300-450-0.5	5	5.8	0.11	5	8.0	0.08
ce300-450-1	5	6.0	0.16	5	7.8	0.16
ce300-450-5	5	1.6	0.85	5	1.6	0.90
ce300-450-10	5	1.2	1.89	5	0.6	2.01
ce300-600-0.5	5	74.6	77.55	5	27083.8	642.41
ce300-600-1	5	34.2	4.44	5	1520.2	2.69
ce300-600-5	5	9.0	2.54	5	53.0	2.29
ce300-600-10	5	7.8	5.01	5	18.6	4.83
ce300-750-0.5	3	19.7	563.75	2	286868.0	2424.76
ce300-750-1	4	35.5	571.38	3	219177.0	758.84
ce300-750-5	5	10.0	14.76	5	36848.8	51.80
ce300-750-10	5	6.2	13.88	5	16440.4	34.18
ce300-900-0.5	2	49.0	3627.58	0		
ce300-900-1	2	43.0	576.39	0		
ce300-900-5	5	16.8	189.19	5	343663.8	1367.85
ce300-900-10	5	6.0	62.53	4	10179.0	36.56
ce400-600-0.5	5	16.6	0.65	5	9.8	0.27
ce400-600-1	4	6.5	0.48	5	11.4	0.68
ce400-600-5	5	9.0	3.0	5	4.0	2.8
ce400-600-10	5	6.8	6.59	5	4.2	6.3
ce400-800-0.5	5	72.8	251.05	3	14283.0	43.05
ce400-800-1	5	13.4	9.64	5	5379.4	19.66
ce400-800-5	5	135.4	521.72	5	2758.2	14.03
ce400-800-10	5	36.4	26.24	5	671.8	18.65
ce400-1000-0.5	1	35	2132.30	0		
ce400-1000-1	0			0		
ce400-1000-5	3	28.0	1423.61	1	365452.0	6805.45
ce400-1000-10	4	41.3	459.72	3	242519.3	3880.49
ce500-750-0.5	5	18.8	3.60	5	30.0	1.85
ce500-750-1	5	31.8	2.20	5	49.8	1.30
ce500-750-5	5	2.8	7.5	5	11.0	7.60
ce500-750-10	5	3.0	17.26	5	11.8	17.41
ce500-1000-0.5	2	100.5	870.07	0		
ce500-1000-1	3	9.7	49.53	2	144505.5	4038.68
ce500-1000-5	5	51.4	186.47	3	30126.3	114.47
ce500-1000-10	5	16.8	57.49	5	22447.8	137.30

Tableau 3.4 – Comparaison de deux méthodes basées sur F1 sur les instances avec $r = 200\,\mathrm{m}$

de nœuds sur l'arbre de recherche dans l'algorithme de branchement et coupes. En fin, la colonne $Temps$ montre le temps de calcul moyen des algorithmes (en secondes).

Comme on peut remarquer au tableau 3.3, pour $t = \{5, 10\}$, notre algorithme est capable de résoudre toutes, sauf l'une des instances jusqu'à 1250 arcs. Il a également été capable de résoudre certains cas avec 1500 arcs. Mais quand t est plus petit, les instances deviennent plus difficiles. Ce comportement contre-intuitif peut être expliqué comme suit. Lorsque le nombre de clients diminue, le nombre d'arcs qui doivent être activés pour assurer la couverture diminue également. Ainsi, il y a plus de combinaisons possibles pour relier ces arcs. Plus d'itérations de PLNE dans l'algorithme de plan sécant ou plus de nœuds sur l'arbre de recherche de l'algorithme de branchement et coupes sont requis pour résoudre le problème. Nous observons également que, pour un nombre donné de nœuds, plus grand est le degré de nœud, plus difficile est l'instance. Cette observation est bien connue pour les problèmes de tournées de véhicules sur arcs.

Au tableau 3.4, nous voyons les résultats pour le case où $r = 200$ mètres. La performance de nos algorithmes dégrade car il ne peut que résoudre les instances avec jusqu'à 1000 arcs (C'est aussi la raison que nous présentons les résultats juste pour ces instances). Lorsque le rayon de couverture augmente, chaque client peut être couvert par plus d'arcs (voir le tableau 3.2), l'espace de solutions augmente et la solution optimale devient plus difficile à trouver.

Les tableaux montrent clairement que l'algorithme de branchement et coupes est moins bon que l'algorithme plan sécant en terme de nombre d'instances résolues. Par rapport à l'algorithme de plan sécant, la qualité de l'algorithme de branchement et coupes diminue rapidement quand la taille du problème augmente. Cela confirme encore une fois l'impact négatif du grand M dans des algorithmes de branchement et coupes.

Comparaison des méthodes exactes pour l'ensemble des formulations

Dans ce paragraphe, nous comparons la performance des algorithmes exacts proposés pour le CEARP. Les cinq algorithmes suivants sont abordés : trois algorithmes de branchement et coupes basés sur les trois formulations F1–F3, l'algorithme de plan sécant basé sur F1 et l'algorithme de branchement et coupes basé sur F2 de Drexl [19]. L'évaluation est réalisée sur les instances non orientées, les instances mixtes et les instances orientées les plus grandes avec 500 nœuds et 1500 arcs (r=150 m), et 500 nœuds et 1000 arcs (r=200 m).

Survol des comparaisons

Le tableau 3.5 compare les cinq algorithmes exacts pour le CEARP en terme de nombre d'instances résolues avec succès à l'optimalité. La colonne CP-F1 donne les résultats pour l'algorithme de plan sécant, et les colonnes B&C-F1, B&C-F2, et B&C-F3 présentent les résultats pour nos trois algorithmes de branchement et coupes basés sur les formulations correspondantes. La colonne Drexl-F2 présente le nombre d'instances réussies pour l'algorithme proposé dans Drexl [19] utilisant CPLEX 11.2 au lieu de 9.1. Les différences les plus importantes entre Drexl-F2 et nos algorithmes (B&C-F1, B&C-F2 et B&C-F3) résident dans la façon de séparer les contraintes de connectivité et dans l'utilisation des contraintes de dominance (Voir l'article 2 à la fin de ce chapitre pour plus d'information). Les formulations sont testées sur les instances orientées.

Instance	CP-F1	B&C-F1	B&C-F2	B&C-F3	Drexl-F2
ce500-1500-0.5	0	0	0	0	0
ce500-1500-1	1	0	2	2	0
ce500-1500-5	4	3	5	5	2
ce500-1500-10	5	5	5	5	4
ce500-1000-0.5	2	0	1	3	1
ce500-1000-1	3	2	4	4	2
ce500-1000-5	5	3	5	5	4
ce500-1000-10	5	5	5	5	5

Tableau 3.5 – Nombre d'instances réussies

A partir du tableau 3.5, une observation intéressante est que les algorithmes de branchement et coupes B&C-F1 et Drexl-F2 sont moins bons que l'algorithme de plan sécant. Nous observons également que la performance des algorithmes B&C-F2 et B&C-F3 est meilleure que celle des autres. Entre B&C-F2 et B&C-F3, pour $r = 150$ m, il semble que B&C-F2 et B&C-F3 sont équivalentes. Pour $r = 200$ m, B&C-F3 est meilleure car elle peut résoudre 2 instances avec $t = 0,5$ que B&C-F2 ne peut résoudre. Donc, dans les paragraphes suivants, nous comparons seulement B&C-F2 et B&C-F3 sur les autres critères et les autres types des instances.

Comparaisons détaillées sur les instances orientées

Ce paragraphe donne les résultats pour les algorithmes de branchement et coupes utilisant F2 et F3. Pour chaque formulation, le tableau 3.6 montre le temps de calcul en seconds (Temps), le gap (Gap) et la valeur de solution (Résultat). Les résultats trouvés montrent que le gap pour B&C-F3 est meilleur que celui pour B&C-F2. B&C-F3 trouve aussi de meilleures solutions que B&C-F2. Nous croyons que c'est parce que F2 contient des contraintes de grand M qui conduisent à des difficultés numériques en raison des nombreuses contraintes de connectivité générées. Celles-ci augmentent la taille du modèle, et CPLEX a besoin de plus de temps pour traiter chaque nœud. Par conséquent, il y a moins de chances de trouver de bonnes solutions.

Comparaisons sur les instances non orientées et mixtes

Le tableau 3.7 donne les résultats pour les deux formulations sur les instances non orientées. La colonne "lb1" présente les valeurs des bornes inférieures au nœud racine des algorithmes. B&C-F3 peut résoudre l'instance MADR-7-3 que B& C-F2 ne peut pas. Il est aussi plus rapide sur 22 des 30 instances. Nous pouvons donc conclure que sur les instances non orientées, B&C-F3 est meilleure que B&C-F2. Notons que dans ce cas, toutes les variables de F3 sont binaires et c'est probablement la raison pour laquelle B&C-F3 est plus efficace que B&C-F2.

Le tableau 3.8 présente les résultats pour les instances mixtes. Tous les résultats sont les valeurs moyennes sur cinq instances. F2 peut résoudre plus d'instances que F3, mais encore une fois son écart à l'optimalité (gap) est, dans quelques cas, faible.

Résultats pour les méthodes approchées

Les tableaux 3.9 et 3.10 donnent les résultats pour les algorithmes UB1 et UB2. Dans ces tableaux, M_{CPP} est la valeur de M calculée par la résolution du problème DCPP. Le nom d'instances en caractères gras indique que ces instances ont été résolues à l'optimum par les algorithmes de branchement et coupes. La colonne "Temps B&C-F2" présente le temps de calcul en secondes de l'algorithme de branchement et coupes basé sur F2. La colonne "Gap" affiche l'écart à l'optimalité en pourcentage des bornes supérieures aux meilleures solutions trouvées par les algorithmes

	B&C-F2			B&C-F3		
Instance	Temps	Gap	Résultat	Temps	Gap	Résultat
ce500-1500-150-0.5-1	7202.29	4.21	106474.2	7202.77	1.76	105231.4
ce500-1500-150-0.5-2	7202.72	3.34	97900.6	7203.01	2.13	97435.3
ce500-1500-150-0.5-3	7202.46	0.82	112363.3	7202.91	0.39	112129.6
ce500-1500-150-0.5-4	7202.45	2.23	95657.7	7202.98	2.53	95778.1
ce500-1500-150-0.5-5	7202.51	2.35	102603.2	7202.91	0.98	102323.9
ce500-1500-150-1-1	7206.93	1.25	129790.7	7207.52	0.40	129659.7
ce500-1500-150-1-2	505.35	0	123123.0	278.86	0	123123.0
ce500-1500-150-1-3	121.73	0	133418.3	598.91	0	133418.3
ce500-1500-150-1-4	7206.53	2.11	116531.6	7207.07	1.04	116058.8
ce500-1500-150-1-5	7206.48	2.21	117967.2	7206.98	1.55	117003.4
ce500-1500-150-5-1	452.86	0	162097.8	278.01	0	162097.8
ce500-1500-150-5-2	128.71	0	160792.7	101.33	0	160792.7
ce500-1500-150-5-3	166.73	0	177242.4	122.72	0	177242.4
ce500-1500-150-5-4	64.19	0	151852.9	78.69	0	151852.9
ce500-1500-150-5-5	147.09	0	161833.4	191.63	0	161833.4
ce500-1500-150-10-1	152.53	0	174504.1	174.82	0	174504.1
ce500-1500-150-10-2	212.81	0	173404.5	190.57	0	173404.5
ce500-1500-150-10-3	197.64	0	185330.8	201.41	0	185330.8
ce500-1500-150-10-4	201.64	0	162071.7	222.80	0	162071.7
ce500-1500-150-10-5	945.25	0	168734.3	240.02	0	168734.3
ce500-1000-200-0.5-1	7200.69	1.59	76033.0	4275.83	0	76033.0
ce500-1000-200-0.5-2	7200.78	1.22	84311.6	385.83	0	84237.2
ce500-1000-200-0.5-3	7200.74	12.82	95790.2	7200.95	0.80	89653.5
ce500-1000-200-0.5-4	7200.71	2.51	76226.2	7201.04	1.52	76084.4
ce500-1000-200-0.5-5	137.05	0	85097.0	1714.54	0	85097.0
ce500-1000-200-1-1	461.58	0	82687.1	4440.20	0	82687.1
ce500-1000-200-1-2	140.30	0	89896.5	257.57	0	89896.5
ce500-1000-200-1-3	6209.50	0	98051.8	279.85	0	98051.8
ce500-1000-200-1-4	7202.02	1.95	82682.3	7202.20	0.41	82344.2
ce500-1000-200-1-5	38.03	0	91915.6	59.10	0	91915.6
ce500-1000-200-5-1	130.37	0	100395.3	1233.24	0	100395.3
ce500-1000-200-5-2	36.33	0	109318.5	92.57	0	109318.5
ce500-1000-200-5-3	121.71	0	114362.3	121.69	0	114362.3
ce500-1000-200-5-4	57.89	0	103790.9	75.01	0	103970.9
ce500-1000-200-5-5	19.58	0	112625.0	52.97	0	112625.0
ce500-1000-200-10-1	67.97	0	110527.9	119.54	0	110527.9
ce500-1000-200-10-2	57.21	0	113694.1	83.24	0	113694.1
ce500-1000-200-10-3	82.84	0	126659.9	95.27	0	123733.9
ce500-1000-200-10-4	69.35	0	115281.4	54.81	0	115281.4
ce500-1000-200-10-5	44.56	0	128163.2	58.86	0	128163.2

Tableau 3.6 – Comparaison de la performance de B&C-F2 et B&C-F3

	B&C-F2					B&C-F3				
Instance	lb1	Gap	BB	Temps	Résultat	lb1	Gap	BB	Temps	Résultat
ALBA-3-1	2391.8	0	236	5.34	2511	2319.2	0	209	2.08	2511
ALBA-3-2	2124.7	0	119	4.14	2324	2011.2	0	74	2.15	2324
ALBA-3-3	2012.8	0	74	1.88	2155	2070.7	0	30	1.32	2155
ALBA-3-4	2363.7	0	959	53.42	3074	2621.0	0	334	13.39	3074
ALBA-3-5	2244.0	0	71	3.63	2440	2355.1	0	21	2.04	2440
ALBA-5-1	2845.0	0	703	28.15	3125	2712.3	0	1105	24.26	3125
ALBA-5-2	2468.3	0	637	24.70	2926	2480.5	0	1655	35.78	2926
ALBA-5-3	3013.8	0	279	9.93	3170	2961.5	0	446	13.12	3170
ALBA-5-4	2447.8	0	298	8.78	2584	2405.8	0	433	7.77	2584
ALBA-5-5	2472.4	0	477	24.66	2642	2535.6	0	112	3.22	2642
ALBA-7-1	2989.7	0	495	21.33	3397	2879.4	0	528	13.64	3397
ALBA-7-2	3200.1	0	793	29.71	3558	3192.1	0	740	21.77	3558
ALBA-7-3	3234.6	0	1939	68.88	3647	3125.3	0	1913	64.31	3647
ALBA-7-4	3255.8	0	365	15.20	3461	3264.4	0	789	24.63	3461
ALBA-7-5	2540.8	0	401	23.27	2821	2642.5	0	249	8.81	2821
MADR-3-1	2511.2	0	11215	2027.39	2925	2658.0	0	4757	137.62	2925
MADR-3-2	3103.3	0	10653	1972.94	3665	3197.5	0	28755	2731.95	3665
MADR-3-3	2580.7	0	2556	366.50	3045	2714.9	0	2516	112.25	3045
MADR-3-4	2829.3	0	8838	2463.51	3295	2936.4	0	11386	794.31	3295
MADR-3-5	2704.7	0	3603	552.06	3165	2760.4	0	7960	1303.2	3165
MADR-5-1	3479.6	0	7116	1398.53	3945	3568.7	0	4261	593.94	3945
MADR-5-2	4234.9	0	3124	280.35	4570	4239.4	0	6871	213.86	4570
MADR-5-3	3850.6	0	17362	4067.07	4505	3810.2	0	4395	480.13	4505
MADR-5-4	3780.9	0	1526	184.09	4020	3840.0	0	2605	180.52	4120
MADR-5-5	3583.5	0	7894	1156.25	4010	3695.5	0	6199	738.96	4010
MADR-7-1	4287.3	0	5041	850.95	4645	4329.4	0	14985	2512.30	4645
MADR-7-2	4363.8	0	7424	1499.57	4650	4350.3	0	4825	477.29	4650
MADR-7-3	4120.9	1.05	33074	7200.08	4620	4234.8	0	16184	2128.87	4620
MADR-7-4	4190.7	2.27	27065	7200.07	4655	4186.7	2.48	39664	7200.06	4645
MADR-7-5	4370.1	0	12706	2958.65	4735	4338.2	0	23533	4535.04	4735

Tableau 3.7 – Comparaison de la performance de B&C-F2 et B&C-F3 sur les instances non orientées

	B&C-F2					B&C-F3				
Instance	Succ	Gap	BB	Temps	Résultat	Succ	Gap	BB	Temps	Résultat
MB0537-0.5	4	1.75	10330.2	3379.42	17592.0	0	1.07	38679.6	7200.72	17460.4
MB0537-1	3	0.59	10425.2	4707.06	18646.8	0	1.33	63897.2	7201.54	18671.4
MB0537-5	5	0	3242.0	185.89	21712.2	3	0.19	138095.6	3937.83	21712.2
MB0537-10	5	0	1040.2	74.01	22666.2	5	0	163733.6	2418.27	22666.2
MB0547-0.5	0	9.28	5450.2	7200.63	15559.8	0	6.95	19457.4	7201.09	15489.2
MB0547-1	0	3.38	14069	7201.74	17155.0	0	3.48	42625.4	7202.20	17216.2
MB0547-5	5	0	4428.6	470.27	21343.2	4	0.13	27233.8	1639.97	21343.2
MB0547-10	5	0	1461.6	79.97	22404.0	5	0	31992.4	756.16	22404.0

Tableau 3.8 – Comparaison de la performance de B&C-F2 et B&C-F3 sur les instances mixtes

de branchement et coupes dans le paragraphe précédent. Les résultats négatifs impliquent que les algorithmes de borne supérieure trouvent les solutions meilleures que les algorithmes de branchement et coupes. A partir de résultats des algorithmes exacts, nous observons qu'un arc est rarement traversé plus de cinq fois dans la solution. Pour cela, nous testons aussi l'algorithme constructif basé sur PLNE (UB2) avec $M = 5$. Notons que cet algorithme n'est pas compétitif sur les instances non orientées parce que dans ces cas nous pouvons borner M efficacement. Lorsque M est bornée plus strictement, F2 est nettement meilleure que F3 ; nous ne présentons donc pas les résultats pour l'algorithme basé sur F3.

Les résultats pour les instances orientées et mixtes montrent la bonne performance de l'algorithme UB2. Il trouve non seulement de meilleures solutions, mais aussi plus rapidement dans presque tous les cas que les algorithmes de branchement et coupes ne peuvent résoudre exactement. La performance de l'algorithme avec $M = 5$ est légèrement meilleure qu'avec $M = M_{CPP}$ mais la limite de M trop stricte peut rendre le problème non réalisable (les instances ce500-1500-5-3-150 et ce500-1500-10-3-150 par exemple). La qualité des solutions de UB1 est beaucoup moins bonne, particulièrement sur les instances orientées où $r = 200\,\text{m}$ ainsi que sur les instances mixtes. UB1 donne de meilleurs résultats seulement sur les instances faciles avec $r = 150\,\text{m}$ et $t = 0,5$ et 1. Cependant, il est beaucoup plus rapide que UB2.

3.4 Conclusion

Dans ce chapitre, nous avons formulé et résolu le CEARP. Deux nouvelles formulations ont été proposées et comparées avec une formulation dans la littérature en se basant sur la performance des algorithmes de branchement et coupes, de plans sécants qui ont été développés pour la résolution exacte du problème. Par rapport à la formulation dans la littérature (F2), l'une des nôtres (F1) a moins de variables et contraintes et l'autre (F3) a un avantage important : elle ne requiert pas les contraintes de grand M. Les résultats montrent que les algorithmes de branchement et coupes basés sur F2 et F3 sont meilleurs que les autres. En comparaison avec F2, notre nouvelle formulation F3 est meilleure sur les instances orientées et non orientées, mais moins bonne sur les instances mixtes. Normalement, F3 donne un meilleur écart à l'optimalité, mais un énorme arbre de séparation et évaluation est le prix à payer. Nous proposons également une heuristique et un algorithme constructif basé sur la PLNE pour la résolution approchée du problème.

En terme de complexité du problème, les instances deviennent plus difficile si le rayon de couverture augmente et si la densité de clients dans le réseau diminue. Nos méthodes exactes sont capable de résoudre les instances orientées assez grandes (jusqu'à 1500 arcs) mais leur performance est encore limite pour les instances mixtes et non orientées. Les méthodes approchées UB1 et UB2 requièrent des améliorations. L'algorithme UB1 est rapide mais la qualité de la solution est encore insuffisante. Au contraire, l'algorithme UB2 donne des solutions très bonnes mais il n'est pas de complexité polynomiale. Deux directions de recherche dans la future devraient être effectuées :

– extension des méthodes proposées pour résoudre un problème plus général avec plusieurs véhicules mCEARP en considérant la contrainte de capacité sur la longueur de chaque route ;

Instance	UB1		UB2-F2, M=5		UB2-F2, M=M_{CPP}			Temps
	Temps	Gap	Temps	Gap	M_{CPP}	Temps	Gap	B&C-F2
ce500-1500-150-0.5-1	6.93	14.71	3819.53	-0.32	11	1128.79	-0.32	7202.29
ce500-1500-150-0.5-2	7.49	16.97	473.07	-0.69	12	1084.34	-0.69	7202.72
ce500-1500-150-0.5-3	7.80	15.81	141.14	-0.02	15	171.44	-0.02	7202.46
ce500-1500-150-0.5-4	8.28	13.30	1337.99	-0.80	12	7202.44	-0.79	7202.45
ce500-1500-150-0.5-5	7.30	13.97	1699.12	-0.33	10	7202.37	-0.33	7202.51
ce500-1500-150-1-1	10.97	10.02	143.32	-0.07	11	129.40	-0.07	7206.93
ce500-1500-150-1-2	11.25	6.24	83.17	0	12	90.92	0	505.35
ce500-1500-150-1-3	11.50	6.39	153.90	0	15	121.17	0	121.73
ce500-1500-150-1-4	11.60	12.62	480.28	-0.10	12	746.87	-0.10	7206.53
ce500-1500-150-1-5	11.19	11.38	7206.52	-0.12	10	7206.26	0.03	7206.48
ce500-1500-150-5-1	57.61	2.94	140.66	0	11	117.08	0	452.86
ce500-1500-150-5-2	54.70	2.41	133.76	0	12	132.17	0	128.71
ce500-1500-150-5-3	54.15	3.58			15	147.08	0	166.73
ce500-1500-150-5-4	51.24	3.62	93.06	0	12	79.82	0	64.19
ce500-1500-150-5-5	53.93	3.04	147.37	0	10	166.00	0	147.09
ce500-1500-150-10-1	119.36	1.75	143.05	0	11	152.90	0	152.53
ce500-1500-150-10-2	107.62	2.26	164.46	0	12	175.16	0	212.81
ce500-1500-150-10-3	118.05	2.57			15	178.73	0	197.64
ce500-1500-150-10-4	101.88	2.78	175.58	0	12	178.49	0	201.64
ce500-1500-150-10-5	115.54	2.91	250.78	0	10	218.16	0	945.25
ce500-1000-200-0.5-1	4.58	30.25	88.65	0	10	54.29	0	7200.69
ce500-1000-200-0.5-2	4.69	42.00	66.63	0	14	161.62	0	7200.78
ce500-1000-200-0.5-3	4.62	33.06	96.97	0	12	354.75	0	7200.74
ce500-1000-200-0.5-4	4.57	41.92	287.06	-0.17	12	252.59	-0.17	7200.71
ce500-1000-200-0.5-5	4.68	39.59	89.17	0	11	81.70	0	137.05
ce500-1000-200-1-1	5.78	24.22	78.95	0	10	106.86	0	461.58
ce500-1000-200-1-2	5.87	35.84	73.87	0	14	182.10	0	140.30
ce500-1000-200-1-3	6.07	23.36	180.66	0	12	81.72	0	6209.50
ce500-1000-200-1-4	5.94	30.50	112.85	0	12	358.85	0	7202.02
ce500-1000-200-1-5	5.99	28.21	23.47	0	11	15.51	0	38.03
ce500-1000-200-5-1	19.57	30.18	60.29	0	10	97.54	0	130.37
ce500-1000-200-5-2	17.98	10.98	74.72	0	14	81.14	0	36.33
ce500-1000-200-5-3	21.04	10.41	58.55	0	12	71.51	0	121.71
ce500-1000-200-5-4	21.14	18.55	34.15	0	12	42.40	0	57.89
ce500-1000-200-5-5	21.19	18.32	24.90	0	11	28.32	0	19.58
ce500-1000-200-10-1	39.85	9.79	67.88	0	10	95.28	0	67.97
ce500-1000-200-10-2	35.85	6.63	49.28	0	14	55.72	0	57.21
ce500-1000-200-10-3	42.70	4.45	80.61	0	12	86.33	0	82.84
ce500-1000-200-10-4	42.54	11.91	49.51	0	12	46.20	0	69.35
ce500-1000-200-10-5	43.33	12.98	62.41	0	11	57.18	0	444.56

Tableau 3.9 – Bornes supérieures basées sur F2 pour les instances orientées

	UB1		UB2-F2, M=5		UB2-F2, M=M_{CPP}			Temps
Instance	Temps	Gap	Temps	Gap	M_{CPP}	Temps	Gap	B&C-F2
MB0537-0.5-1	4.77	34.40	345.39	0	7	278.38	0	992.57
MB0537-0.5-2	4.77	39.41	382.67	0	7	397.07	0	4491.65
MB0537-0.5-3	5.04	36.08	2726.26	-0.51	7	981.18	-0.51	7200.69
MB0537-0.5-4	4.73	37.74	146.45	0	7	212.68	0	690.54
MB0537-0.5-5	5.09	31.65	478.33	0	7	214.05	0	3521.67
MB0537-1-1	5.79	33.56	276.76	0	7	155.04	0	1382.62
MB0537-1-2	5.79	33.60	108.31	0	7	127.44	0	3629.55
MB0537-1-3	5.78	33.08	393.87	-0.45	7	287.98	-0.45	7201.13
MB0537-1-4	5.53	38.33	723.45	0	7	3143.52	0	7201.26
MB0537-1-5	5.88	34.87	2471.76	0	7	617.78	0	4120.76
MB0537-5-1	13.40	17.44	274.46	0	7	283.69	0	593.70
MB0537-5-2	13.08	15.59	41.81	0	7	50.01	0	60.67
MB0537-5-3	13.78	18.12	92.07	0	7	96.55	0	138.32
MB0537-5-4	14.10	17.96	63.01	0	7	107.87	0	93.19
MB0537-5-5	13.66	15.07	64.30	0	7	57.08	0	43.58
MB0537-10-1	23.27	13.45	69.48	0	7	72.61	0	73.15
MB0537-10-2	25.68	14.77	67.03	0	7	74.79	0	75.52
MB0537-10-3	25.64	18.29	185.10	0	7	144.82	0	114.87
MB0537-10-4	25.22	11.42	49.43	0	7	30.24	0	30.61
MB0537-10-5	24.12	14.85	40.86	0	7	46.35	0	75.92
MB0547-0.5-1	5.60	44.89	7200.58	-4.03	14	7200.88	-1.40	7200.65
MB0547-0.5-2	5.23	41.12	1460.73	-1.20	14	2645.67	-1.20	7200.60
MB0547-0.5-3	5.22	45.92	7200.57	-1.06	14	2215.78	-1.33	7200.59
MB0547-0.5-4	5.31	39.19	7200.69	-1.34	14	7201.31	-0.65	7200.62
MB0547-0.5-5	5.21	40.64	7200.58	-1.65	14	7200.61	-1.63	7200.68
MB0547-1-1	6.25	36.72	66.54	-0.17	14	233.73	-0.17	7201.69
MB0547-1-2	6.27	33.26	383.36	-0.31	14	1211.10	-0.31	7201.86
MB0547-1-3	6.54	26.39	7201.62	-1.06	14	7201.70	-1.09	7201.60
MB0547-1-4	6.20	32.59	954.23	-0.21	14	969.81	-0.21	7201.87
MB0547-1-5	6.21	33.52	7201.62	-0.69	14	7201.59	-0.66	7201.67
MB0547-5-1	17.35	29.24	17.54	0	14	24.49	0	46.89
MB0547-5-2	17.62	40.39	18.28	0	14	43.18	0	32.89
MB0547-5-3	18.28	139.02	18.05	0	14	67.34	0	2158.19
MB0547-5-4	17.27	14.59	33.64	0	14	50.53	0	40.31
MB0547-5-5	17.46	16.91	64.69	0	14	52.02	0	73.08
MB0547-10-1	22.94	44.51	17.99	0	14	48.91	0	98.00
MB0547-10-2	35.85	14.71	60.13	0	14	67.20	0	54.33
MB0547-10-3	32.65	12.50	83.55	0	14	105.39	0	138.32
MB0547-10-4	31.59	11.56	69.47	0	14	64.96	0	47.86
MB0547-10-5	31.88	14.39	55.85	0	14	67.30	0	61.35

Tableau 3.10 – Bornes supérieures basées sur F2 pour les instances mixtes

– construction de bonnes méthodes approchées de complexité polynomiale pour traiter les instances réelles du CEARP.

Article 1 : An Exact Algorithm for the Close Enough Traveling Salesman Problem with Arc Covering Constraints

Minh Hoàng Hà, Nathalie Bostel, André Langevin, Louis-Martin Rousseau (2012). *An Exact Algorithm for the Close Enough Traveling Salesman Problem with Arc Covering Constraints.* In Calors J. Luz and Fernando Valente (Eds) Proceedings of the 1st International Conference on Operations Research and Enterprise Systems (ICORES), Vilamoura, Algavre, Portugal, 4-6 February, 2012, pp. 233-239, ISBN : 978-989 8425-97-3.

AN EXACT ALGORITHM FOR THE CLOSE ENOUGH TRAVELING SALESMAN PROBLEM WITH ARC COVERING CONSTRAINTS

Minh Hoang Ha[1,3], Nathalie Bostel[2], André Langevin[1] and Louis-Martin Rousseau[1]

[1]*Department of Mathematics and Industrial Engineering and CIRRELT, École Polytechnique de Montréal, C.P. 6079, Succursale Centre-ville, Montréal, Qué. Canada H3C 3A7*

[2]*L'UNAM Université, Université de Nantes, IRCCyN UMR CNRS 6597, 58 rue Michel Ange, B.P. 420, 44606 Saint-Nazaire Cedex, France*

[3]*L'UNAM Université, École des Mines de Nantes, IRCCyN UMR CNRS 6597, 4 rue Alfred Kastler, 44307 Nantes Cedex 3, France*

Keywords: Close enough traveling salesman problem, automated meter reading, radio frequency identification, cutting planes.

Abstract: In this paper, we consider a problem still seldom studied in the literature, the Close Enough Traveling Salesman problem with covering constraints on the arcs. This problem arises in the context of utility companies that use automated meter reading (AMR) with radio frequency identification (RFID) technology to read meters from a distance. The contribution of this paper is to introduce a new mathematical formulation and to propose a first exact algorithm for this problem. Computational results show that our algorithm is capable of solving to optimality instances of realistic size, such as those introduced in (Golden et al., 2008), with 1000 arcs and 9000 customers in less than 2 hours.

1 INTRODUCTION

Recent technological advances in the combined usage of automated meter reading and radio frequency identification are allowing an increasing number of utility companies to retrieve customer information remotely. With this technology, the exact consumption of a resource (electricity, gas or water) by customers is coded in a unique radio frequency signal and transmitted from metering devices, situated at the customer location, to portable receivers, which can be mounted in moving vehicles. The effective radius of automated meter reading (AMR), also called read range, is normally between 150 and 300 meters, but may be as high as 381 meters (see (Golden et al., 2008)). Although AMR is based on various technologies, we consider the most common version (radio frequency based) to study a variant of an arc routing problem dealing with mobile or "drive-by" meter reading. In this variant, a reading device is installed in a vehicle, which drives around to collect remotely the data sent by metering devices. Thus, the reader does not have to reach each customer to collect the data, but only needs to pass within its read range. Each vehicle must traverse a service area and pass close enough to each meter so that they can all be read.

More formally, let $G = (V,A)$ be a directed graph, where V is the vertex set, $V = \{v_0, ..., v_{n-1}\}$, vertex v_0 is the depot, and $A = \{(v_i, v_j) : v_i, v_j \in V\}$ is the arc set of size m. Numbering the arcs enables A to be expressed as $\{a_1, a_2, ..., a_m\}$ and a cost c_a to be associated with each arc of A. Let $W = \{w_1, w_2, ..., w_l\}$ be a set of customers that must be covered. These customers can be located everywhere on the area covered by the network, and have a different vertex. The *Close Enough Traveling Salesman Problem with arc covering constraints* (CETSP) consists in finding a minimum cost tour, which begins and ends at the depot, such that every customer of W is covered by the tour, i.e. lies within a distance r from an *arc* of the tour. The CETSP is NP-hard as it reduces to a *Rural Postman Problem* (RPP) when $r = 0$, i.e. every customer of W coincides with a point on an arc of A, and the number of arcs containing the customers (called required arcs) is less than m, the number of arcs in the graph. Note that, if the number of required arcs is equal to m, the problem becomes a *Chinese Postman Problem* (CPP).

To the best of our knowledge, the only work on the CETSP can be credited to (Golden et al., 2008). The authors call the problem a *CETSP over a street network* and propose four heuristics to solve 18 instances with an average of about 900 arcs and 9000 customers each. Among these instances, only two of them are described in more detail: a sparse instance that includes 3345 customers and 405 arcs and a dense

instance with 10,230 customers and 1099 arcs. Basically, the heuristics of (Golden et al., 2008) are implemented through a two-stage process. In stage one, the heuristic identifies a subset of arcs to be traversed, either with some simple greedy procedures or through solving an integer program. In stage two, the problem becomes the well-known *Rural Postman Problem* (RPP) and is solved by a sophisticated heuristic.

There is another problem which can be seen as a *CETSP with the vertex covering constraints*. It is named *the covering tour problem* by (Gendreau et al., 1997) and is similar to the CETSP with arc covering constraints except that a closed tour has to be determined so that every vertex of W lies within a distance r from a *vertex* of the tour. In (Gendreau et al., 1997), an exact algorithm and a heuristic are presented. Heuristics have also been proposed for problems belonging to the family of the covering tour problem, such as *the CETSP in the plane* in (Gulczynski et al., 2006) and (Dong et al., 2007).

In this paper, we address the CETSP, which we formulate as an integer program and solve through a cutting plane approach. Computational experiments show that our approach is efficient, being capable of solving instances in some cases with up to 1500 arcs in less than 2 hours.

The remainder of the paper is organized as follows: Section 2 presents in detail our exact algorithm to solve the CETSP. Computational results are reported and analysed in Section 3. Finally, Section 4 summarizes our conclusions.

2 EXACT ALGORITHM FOR CETSP

Given a node subset, $S \subseteq V$, let $\delta^+(S)$ denote the set of outgoing arcs of S and $\delta^-(S)$ denote the set of incoming arcs of S. If $S = \{v_k\}$, we simply write $\delta^+(k)$ (or $\delta^-(k)$) instead of $\delta^+(\{v_k\})$ (or $\delta^-(\{v_k\})$). $E(S)$ is the set of arcs with both end-points in S. Let x_a be the number of times arc a is traversed, and c_a the associated cost (distance or travel time). We define the binary coefficients λ_{lk} equal to 1 if and only if $w_l \in W$ can be covered by $a_k \in A$. Given $x \in N^{|A|}$ and $T \subset A$, $x(T)$ denotes $\sum_{e \in T} x_e$. Then the CETSP can be stated as:

$$\text{Minimize} \sum_{a \in A} c_a x_a \tag{1}$$

$$\text{subject to } x\Big(\delta^+(0)\Big) \geq 1 \tag{2}$$

$$x\Big(\delta^+(i)\Big) - x\Big(\delta^-(i)\Big) = 0 \; \forall i \in V \tag{3}$$

$$\sum_{a \in A} x_a.\lambda_{wa} \geq 1 \; \forall w \in W \tag{4}$$

$$Mx\Big(\delta^+(S)\Big) - x\Big(E(S)\Big) \geq 0 \; \forall S \subset V - \{v_0\} \text{ and } 2 \leq |S| \leq n-2 \tag{5}$$

$$x_a \in Z^+ \; \forall a \in A \tag{6}$$

where M is a large number. The objective (1) is to minimize the total travel cost. Constraint (2) ensures that the depot belongs to the tour, while (3) are the flow conservation constraints. Constraints (4) enforce that every customer of W is covered by the tour and constraints (5) are the disjoint subtour elimination constraints. (In arc routing, an optimal tour can contain a cycle. The disjoint "subtour elimination constraints" eliminate subtours disconnected from the tour containing the depot). These constraints force the presence of at least one outgoing arc of any set S, for every possible subset S of V containing an arc belonging to the tour. Constraints (6) define the variable domains.

We propose to solve the CETSP optimally through a cutting plane approach. The main strategy of this algorithm is to solve iteratively an integer program including the constraints (2), (3), (4), and (6). At each iteration, the disjoint subtour elimination constraints (5) violated by the optimal solution are added to the model. A disjoint subtour can be identified through a depth-first search which, starting from a given vertex, traverses all the solution arcs to reach all other vertices. The encountered vertices marked are placed on a stack in the order in which they are visited. After an arc is traversed, it is removed from the current graph. A disjoint subtour is created when, starting from some vertex, it is not possible to mark all the vertices present in the solution. Once a disjoint subtour is identified, the process can be repeated starting from any unmarked vertex of the solution.

Note that the constraints (5) can only be applied to the disjoint subtours that do not contain the depot. This is because, for the subtours S_0 containing the depot, the disjoint subtour elimination can be obtained by the use of arcs not already used in the solution but with head points belonging to S_0, the set of nodes of the subtour containing the depot. In order to improve the performance of the algorithm, we first check the covering of such subtours. If we can not cover all the

customers with the arcs of S_0 (used and unused), the constraint (5) is then still applied; otherwise, we add the following constraint for S_0:

$$Mx\Big(\delta^*(S_0)\Big) - x\Big(E'(S_0)\Big) \geq 0 \qquad (7)$$

where $E'(S_0)$ denotes the set of arcs used in S_0 while $\delta^*(S_0)$ denotes the set of arcs that are not used and whose heads are in S_0.

3 COMPUTATIONAL EXPERIMENTS

In this section, we describe the CETSP instances and the computational evaluation of the proposed approach. Our algorithm is coded in C/C++ using Cplex 12.1 with Callable Library and was run on a 2.4 GHz CPU with 2GB RAM. The running time for each instance was limited to 2 hours. We have tested different values of M (5000, 10,000 and 20,000) and observed that its impact on the performance of the algorithm is negligible. We decided to predetermine M at a value of 10,000 in the implementation.

3.1 Data instances

To build the CETSP instances, we randomly generate graphs that imitate real street networks by the following procedure:

- The coordinates of n vertices are randomly generated in a unitary square. Then a heuristic is used to find the shortest tour passing through all the nodes exactly once. This tour is a Hamiltonian cycle and is used as a framework to construct the full graph. The resulting graph is therefore strongly connected.
- In order to imitate real networks, random arcs are added to the current tour to reach a total number of arcs $m = nd$, where n denotes the number of vertices and d the ratio between the number of arcs and the number of nodes, in such a way that: (i) the arcs are not too long, and (ii) there is no intersection between any two arcs.

In our tests, we use the graphs with the number of vertices $n \in \{300, 400, 500\}$ and the ratio between the number of arcs and the number of nodes $d \in \{1.5, 2, 2.5, 3\}$. For each couple of n and d, we have generated 10 different graphs. Arc costs are defined as c_{ij} kilometres, where c_{ij} is the Euclidean distance between v_i and v_j multiplied by 5 to obtain an average length of arcs close to reality (from about 0.2 to 0.4 kilometres).

Once the graphs are created, the CETSP instances are generated by randomly positioning $q = mt$ customer nodes on the square containing the graph, where m denotes the number of arcs and t the ratio between the number of customers and the number of arcs, $t \in \{0.5, 1, 5, 10\}$. Thus, for each graph, four CETSP instances are created. The effective RFID radius r is set at a value of 150 meters. In order to ensure the existence of a solution, we delete all the customers that can not be covered by any arc. We also examine the impact of increasing the radius parameter from 150 meters to 200 meters. To do this, we still use the graph created with r = 150 meters but change the read range to 200 meters. In other words, the arcs and coordinates of the customers and vertices are kept constant while r is increased.

For each value of r, we thus generate 480 CETSP instances named ce-n-k-t, where n is the number of nodes, k indicates the number of arcs and t is the ratio between the number of customers and the number of arcs. For example, ce-300-450-10 stands for an instance with 300 nodes, 450 arcs, and t =10.

In order to analyse further the impact of customer number on our algorithm, we use a customer reduction procedure. Given $w_l \in W$, let $Z(w_l)$ be the set of arcs that can cover w_l. Consider each pair of customers w_i and w_j, if $Z(w_i) \subseteq Z(w_j)$ then customer w_j can be eliminated. This is because when we service w_i, w_j is covered at the same time. Note that, the number of remaining customers is also the maximum number of arcs that have to be activated for covering purpose.

3.2 Computational results

Tables 1 and 2 present the characteristics of the instances and the computational results obtained for two different values of r (radius of AMR). They include the name of the instance and the average number of remaining customers after the reduction procedure (in columns 1 and 2). $\#ofOpt$ indicates the number of optimal solutions obtained for each set and $OptVal$ the average optimal value (in km) for these solutions. $\#ofIPIter$ presents the average, minimum and maximum number of integer programming iterations. *Time* shows the average, minimum and maximum running time (in seconds).

The results presented on Table 1 and 2 indicate that the ratio between the number of remaining customers and the number of arcs in graph is often between 0.2 and 0.5. Therefore, the number of arcs that must be activated for covering purpose is smaller than the total number of arcs and our instances are thus far from a CPP.

As can be observed in Table 1, for $t = \{5,10\}$ our algorithm was able to solve all but one of the instances with 1250 arcs. It was also capable of solving some instances with up to 1500 arcs. But when t is smaller, the instances become more difficult. This counterintuitive behaviour can be explained as follows. When the number of customers decreases, the number of arcs that must be activated for covering purpose also decreases. Thus, there are more potential combinations to connect these "covering" arcs. More MIP iterations are thus needed to solve the problem.

We also observe that, for a given number of vertices, the greater the vertex degree is, the harder the instance is.

In Table 2, we see the results for the case where $r = 200$ meters. The performance of our algorithm degrades as it can only solve the instances with up to 1000 arcs. When the read range increases, each customer can be covered by more arcs, so the solution space increases and the optimal solution becomes more difficult to find. Obviously, increasing the read range leads to a considerable decrease in the traveling distance and cost. The column *OptVal* confirms this remark. Developing new technology to allow an increased read range could therefore be an effective means to reduce the distance driven to collect customer information.

4 CONCLUSION

In this paper, we have formulated and solved the CETSP with arc covering constraints. An integer linear programming formulation has been proposed and solved through a cutting plane algorithm. Computational results on a set of 960 instances have been reported and analysed. These results show that our algorithm is capable of solving to optimality instances of realistic size and works better when there are many customers to be covered. As we notice that RFID technology is rather used when the customer density is important, our algorithm is quite suitable to solve real problems of utility companies.

REFERENCES

Dong, J., Yang, N., and Chen, M. (2007). *Extending the Horizons: Advances in Computing, Optimisation, and Decision Technologies*, chapter Heuristic Approaches for a TSP Variant: The Automatic Meter Reading Shortest Tour Problem, pages 145–163. Springer Verlag.

Gendreau, M., Laporte, G., and Semet, F. (1997). The covering tour problem. *Operations Research*, 45:568–576.

Golden, B., Raghavan, S., and Wasil, E. (2008). *The Vehicle Routing Problem: Lastest Avances and New Challenges*, chapter Advances in Meter Reading: Heuristic Solution of the Close Enough Traveling Salesman Problem over a Street Network, pages 487–501. Springer.

Gulczynski, D., Heath, J., and Price, C. (2006). *Perspectives in Operations Research: Papers in Honor of Saul Gass 80th Birthday*, chapter The Close Enough Traveling Salesman Problem: A Discussion of Several Heuristics, pages 271–283. Springer Verlag.

Table 1: Computational results with r = 150 meters

Data	# of Cus	# of Opt	Opt Val (km)	# of IP Iter			Time (sec)		
				Aver.	Min	Max	Aver.	Min	Max
ce300-450-0.5	102	10	94.5	6.4	2	24	0.52	0.05	2.5
ce300-450-1	142	10	118.8	2.8	1	8	0.08	0.02	0.31
ce300-450-5	202	10	175.6	1.6	1	3	0.08	0.05	0.16
ce300-450-10	212	10	195.4	1.6	1	3	0.15	0.09	0.36
ce300-600-0.5	126	10	100.6	11.5	2	31	3.98	0.17	14.61
ce300-600-1	166	10	123.5	8.5	2	33	3.01	0.14	17.78
ce300-600-5	224	10	182.1	4.9	2	27	1.07	0.16	8.88
ce300-600-10	236	10	195.6	2.1	1	4	0.27	0.16	0.45
ce300-750-0.5	147	9	95.6	36.2	5	112	308.52	1.98	1807.64
ce300-750-1	193	10	120.4	8.2	4	12	5.40	1.59	10.17
ce300-750-5	248	10	172.1	3.3	2	6	0.70	0.25	2.30
ce300-750-10	264	10	186.3	2.9	2	5	0.60	0.30	1.08
ce300-900-0.5	166	7	99.1	12.6	4	32	194.09	1.77	1034.83
ce300-900-1	213	10	122.0	10.9	2	26	45.03	0.56	258.08
ce300-900-5	276	10	163.5	3.9	2	9	3.24	0.56	11.52
ce300-900-10	292	10	175.0	3.1	1	5	2.18	0.61	3.78
ce400-600-0.5	144	10	100.3	11.2	2	49	1.94	0.05	12.45
ce400-600-1	194	10	129.4	3.3	2	7	0.16	0.06	0.41
ce400-600-5	260	10	185.1	1.8	1	3	0.13	0.08	0.22
ce400-600-10	274	10	207.1	1.5	1	3	0.19	0.14	0.30
ce400-800-0.5	174	9	102.1	35.7	4	88	88.25	1.76	356.92
ce400-800-1	226	10	126.5	17.2	3	42	11.49	0.75	33.45
ce400-800-5	296	10	178.8	3.8	2	12	1.09	0.22	7.03
ce400-800-10	309	10	199.0	2.6	2	5	0.52	0.30	1.64
ce400-1000-0.5	198	3	103.5	11.0	10	13	745.15	40.94	2113.89
ce400-1000-1	252	9	122.1	13.1	5	25	492.43	4.63	1992.3
ce400-1000-5	313	10	169.2	4.9	2	13	4.39	0.72	18.20
ce400-1000-10	327	10	183.5	4.2	2	9	4.94	0.59	24.59
ce400-1200-0.5	221	0							
ce400-1200-1	279	4	123.0	8.0	4	12	73.65	40.03	135.44
ce400-1200-5	357	10	157.1	5.7	4	9	29.07	6.43	129.79
ce400-1200-10	373	10	167.0	6.9	3	23	26.83	3.21	91.48
ce500-750-0.5	185	10	113.5	39.0	3	323	674.82	0.13	6731.81
ce500-750-1	249	10	141.3	10.7	2	70	6.71	0.11	63.13
ce500-750-5	319	10	201.3	2.6	2	4	0.21	0.17	0.28
ce500-750-10	328	10	227.1	2.2	1	4	0.30	0.23	0.43
ce500-1000-0.5	224	6	126.2	58.0	8	211	1271.11	7.09	6161.19
ce500-1000-1	285	10	131.0	14.3	3	65	30.72	0.95	102.42
ce500-1000-5	360	10	181.8	4.4	3	9	1.80	0.57	7.25
ce500-1000-10	375	10	203.0	4.1	2	6	1.47	0.58	3.00
ce500-1250-0.5	262	0							
ce500-1250-1	330	4	122.5	13.8	5	30	1385.46	235.72	2496.89
ce500-1250-5	408	9	172.0	9.0	3	31	62.24	2.17	201.83
ce500-1250-10	428	10	182.2	5.5	3	13	38.53	2.48	124.22
ce500-1500-0.5	284	0							
ce500-1500-1	354	0							
ce500-1500-5	458	5	163.9	8.0	6	12	402.75	32.19	1008.89
ce500-1500-10	477	8	172.0	5.8	4	8	353.63	15.22	1994.36

Table 2: Computational results with r = 200 meters

Data	# of Cus	# of Opt	Opt Val (km)	# of IP Iter			Time (sec)		
				Aver.	Min	Max	Aver.	Min	Max
ce300-450-0.5	101	10	74.2	20.9	2	93	7.76	0.05	52.38
ce300-450-1	143	10	89.3	11.3	2	70	1.31	0.05	10.98
ce300-450-5	197	10	115.7	1.5	3	3	0.07	0.05	0.13
ce300-450-10	205	10	129.5	1.3	2	3	0.11	0.08	0.18
ce300-600-0.5	126	9	74.1	53.3	10	201	844.57	4.42	6959.88
ce300-600-1	168	10	83.5	23.2	4	65	16.74	0.72	60.38
ce300-600-5	217	10	115.1	6.4	2	25	1.91	0.13	8.88
ce300-600-10	220	10	125.7	5.6	1	24	1.72	0.22	11.94
ce300-750-0.5	149	4	71.4	29.3	13	61	2981.45	208.53	7088.2
ce300-750-1	192	7	87.5	29.0	8	109	1749.44	14.36	7198.27
ce300-750-5	238	10	113.9	11.2	5	37	31.05	2.41	145.95
ce300-750-10	240	10	122.9	7.0	4	12	7.83	1.88	22.86
ce300-900-0.5	169	1	81.4	87.0	87	87	4352.8	4352.8	4352.8
ce300-900-1	212	3	90.0	35.7	9	52	2895.48	127.5	7070.88
ce300-900-5	264	9	110.4	15.4	4	55	761.51	6.25	2542.66
ce300-900-10	269	10	117.2	8.1	3	25	148.26	2.80	1225.56
ce400-600-0.5	146	10	79.2	13.4	2	50	4.37	0.08	28.67
ce400-600-1	202	9	88.3	22.4	1	157	24.17	0.06	209.91
ce400-600-5	274	10	114.3	6.4	2	21	0.90	0.13	6.06
ce400-600-10	276	10	123.6	4.9	1	25	0.52	0.20	2.66
ce400-800-0.5	180	7	77.6	20.0	7	43	89.64	6.02	264.20
ce400-800-1	239	8	85.2	11.3	6	25	18.18	2.77	62.84
ce400-800-5	314	9	111.6	8.8	4	22	6.61	1.44	16.95
ce400-800-10	321	10	120.3	24.5	4	162	180.89	1.22	1723.02
ce400-1000-0.5	208	0							
ce400-1000-1	270	1	95.9	24.0	24	24	7171.13	7171.13	7171.13
ce400-1000-5	331	0							
ce400-1000-10	339	6	116.0	16.0	6	39	1874.72	18.91	7186.65
ce500-750-0.5	192	10	78.0	51.8	3	326	384.42	0.36	3534.50
ce500-750-1	263	10	96.3	28.1	3	181	54.15	0.25	505.33
ce500-750-5	363	10	118.6	5.0	2	24	1.44	0.19	11.27
ce500-750-10	377	10	128.8	3.0	2	4	0.54	0.34	0.78
ce500-1000-0.5	242	1	85.1	22.0	22	22	842.83	842.83	842.83
ce500-1000-1	321	5	90.8	19.8	9	52	436.34	13.56	814.22
ce500-1000-5	411	9	109.0	28.2	6	78	1025.59	6.23	5406.33
ce500-1000-10	429	8	117.8	17.3	5	28	483.14	2.81	3248.66

Article 2 : Solving the Close-Enough Arc Routing Problem

accepté pour publication dans la revue internationale *Networks*.

Solving the Close-Enough Arc Routing Problem

Minh Hoang Ha[a,c], Nathalie Bostel[b], André Langevin[c], Louis-Martin Rousseau[c,*]

[a] *L'UNAM Université, École des Mines de Nantes, IRCCyN UMR CNRS 6597, 4 rue Alfred Kastler, 44307 Nantes Cedex 3, France*
[b] *L'UNAM Université, Université de Nantes, IRCCyN UMR CNRS 6597, 58 rue Michel Ange, B.P. 420, 44606 Saint-Nazaire Cedex, France*
[c] *Department of Mathematics and Industrial Engineering and CIRRELT, École Polytechnique de Montréal, C.P. 6079, Succursale Centre-ville, Montréal, QC, Canada H3C 3A7*

Abstract

The close-enough arc routing problem (CEARP) has an interesting real-life application to routing for meter reading. In this article, we propose a new mathematical formulation for this problem. We analyze our formulation and compare it with two formulations in the literature. We also develop branch-and-cut algorithms to solve the problem to optimality. We present computational results for instances based on three types of graphs: directed, undirected, and mixed.

Keywords: close-enough arc routing problem, close-enough traveling salesman problem, automated meter reading, radio frequency identification, branch-and-cut algorithm

1. Introduction

Consider a directed graph $G = (V, A)$, with vertex set $V = \{v_0, v_1, ..., v_{n-1}\}$, and arc set of size m, $A = \{(v_i, v_j) : v_i, v_j \in V\}$. Numbering the arcs allows A to be expressed as $\{a_1, a_2, ..., a_m\}$. Let c_a be the cost associated with each arc of A. The well-known directed rural postman problem (DRPP) is the problem of determining a minimum-cost closed route traversing each arc in $A_r \subset A$ (called required arcs) at least once. Several algorithms have been

*Corresponding author: Tel.: (+1)514-340-4711 # 4569; fax:(514)340-4463
Email address: louis-martin.rousseau@polymtl.ca (Louis-Martin Rousseau)

proposed to solve the DRPP (see [1], [2], and [3]). The closed-enough arc routing problem (CEARP) is a generalization of the DRPP in which the subset of required arcs A_r is not defined. Instead, there is a set of customers $W = \{w_1, w_2, ..., w_l\}$ that must be covered. These customers can be located anywhere in the area covered by the network, not only in the network itself. As in the original version of this problem (see [4]), we consider the inclusion of a depot. By definition, vertex v_0 is this depot. The CEARP consists in finding a minimum-cost tour, which begins and ends at the depot, such that every customer of W is covered by the tour, i.e., lies within a distance r of an *arc* of the tour.

The main application of this problem is to construct routes for meter reading (see [4] and [5] for further information). In this application, to measure customers' consumption information from a distance, a reading device is installed in a vehicle to collect the data sent by metering devices. The reader does not have to visit each customer to collect the data but must enter the meter's read range. The vehicle must traverse the service area and pass close enough to each meter so that they can all be read. The effective radius r, also called the read range, is normally between 150 and 300 m but may be as high as 381 m (see [4]).

The CEARP is clearly NP-hard because it contains the DRPP as a special case. The existing literature on the CEARP is limited. The earliest work on the CEARP is [4]. The authors call the problem a *close-enough traveling salesman problem (CETSP) over a street network* and propose four heuristics to solve eighteen real-life instances with an average of about 900 street segments and 9000 customers each. The read range r is tested with two values: about 100 and 150 m, respectively. Basically, the heuristics of [4] are implemented in a two-stage process. In stage one, the heuristic identifies a subset of arcs to be traversed, either with some simple greedy procedures or by solving an integer program. In stage two, the problem becomes the well-known DRPP and is solved by a sophisticated heuristic.

To the best of our knowledge, the only exact method for the CEARP is [5]. The authors propose an initial formulation for the problem in which there is a connectivity constraint, and they solve this optimally using a cutting-plane approach. The algorithm first solves the problem without the connectivity constraints. The violated connectivity constraints are then added to the model, and the process is repeated until the connectivity is satisfied. Computational results show that this algorithm can solve to optimality random instances of a realistic size, such as those introduced in [4].

The CEARP is equivalent to the generalized directed rural postman problem (GDRPP). In the GDRPP, there are several subsets of arcs (also called clusters) and the objective is to find a minimum-cost tour traversing at least one arc from each cluster. The clusters may be connected or disjoint. In the CEARP, the arcs covering a customer correspond to a subset in the GDRPP. The version of the GDRPP with no depot is described in [6], where both exact and heuristic methods are presented. The exact method is a branch-and-cut algorithm based on a mathematical formulation.

There is another problem that can be seen as a *CETSP with vertex-covering constraints*. It is the *covering tour problem* (CTP) [7] and is similar to the CEARP except that a closed tour has to be determined so that every vertex of W lies within a distance r of a *vertex* of the tour. In [7], an exact algorithm and a heuristic are presented. A heuristic based on the scatter-search method has been proposed in [8]. Heuristics for the multi-vehicle version (m-CTP) are presented in [9]. Heuristics have also been proposed for problems belonging to the family of covering tour problems, such as *the CETSP in the plane*; see [10] and [11].

The aim of this paper is to design exact algorithms for the CEARP. Our main contributions are: 1) we introduce a new formulation for the CEARP; 2) we compare, both analytically and empirically, this formulation with two others, the first introduced in [5] and the second presented in [6] for the GDRPP; 3) we improve the branch-and-cut algorithm of [6] and propose a new algorithm for the CEARP; and 4) we propose a MIP-based constructive algorithm to solve the CEARP in practice.

The remainder of the paper is organized as follows. Section 2 presents in detail our branch-and-cut algorithms and a MIP-based constructive algorithm to solve the CEARP. The computational results are reported and analyzed in Section 3. Finally, Section 4 summarizes our conclusions.

2. Branch-and-cut algorithms for CEARP

2.1. Mathematical formulations

Given a node subset, $S \subseteq V$, let $\delta^+(S)$ denote the set of outgoing arcs of S and $\delta^-(S)$ denote the set of incoming arcs of S. If $S = \{v_k\}$, we simply write $\delta^+(k)$ (or $\delta^-(k)$) instead of $\delta^+(\{v_k\})$ (or $\delta^-(\{v_k\})$). $A(S)$ is the set of arcs with both endpoints in S. Each arc a is associated with a cost c_a (distance or travel time). We define the binary coefficients λ_{lk} to be equal

to 1 if and only if $w_l \in W$ can be covered by $a_k \in A$. Given $x \in N^{|A|}$ and $T \subset A$, $x(T)$ denotes $\sum_{e \in T} x_e$.

In [5], the following formulation, denoted by F1, was presented:

$$\text{F1:} \qquad \text{Minimize} \sum_{a \in A} c_a x_a \tag{1}$$

$$\text{subject to} \qquad x\Big(\delta^+(0)\Big) \geq 1 \tag{2}$$

$$x\Big(\delta^+(i)\Big) - x\Big(\delta^-(i)\Big) = 0 \qquad \forall i \in V \tag{3}$$

$$\sum_{a \in A} \lambda_{wa} x_a \geq 1 \qquad \forall w \in W \tag{4}$$

$$Mx\Big(\delta^+(S)\Big) - x_a \geq 0 \qquad \forall S \subset V - \{v_0\}$$
$$\text{and } 2 \leq |S| \leq n-2,\ a \in A(S) \tag{5}$$

$$x_a \in Z^+ \qquad \forall a \in A \tag{6}$$

where M is a large number representing an upper bound on the number of times an arc is used. As in [6], M is set to $|A| + 1$.

The objective (1) minimizes total travel cost. Constraint (2) ensures that the depot belongs to the tour, and constraints (3) are the flow conservation constraints. Constraints (4) ensure that every customer of W is covered by the tour, and constraints (5) are the connectivity constraints. These constraints ensure the presence of at least one outgoing arc of any set S, for every possible subset S of V containing an arc belonging to the tour. Constraints (6) define the variables.

Since the CEARP is equivalent to the GDRPP, the formulation in [6] proposed for the GDRPP can be used for the CEARP. Because the formulation in [6] addresses the no-depot version of the problem, we modify it slightly. In this formulation, y_i is a binary variable that indicates the use of vertex i in the solution, and the integer variable x_a denotes the number of times that arc a is traversed. Let he_a be the head of arc a. The formulation for the CEARP is as follows:

F2: Minimize $$\sum_{a \in A} c_a x_a \tag{7}$$

subject to
$$y_0 = 1 \tag{8}$$
$$x\Big(\delta^+(i)\Big) - x\Big(\delta^-(i)\Big) = 0 \qquad \forall i \in V \tag{9}$$
$$\sum_{a \in A} \lambda_{wa} x_a \geq 1 \qquad \forall w \in W \tag{10}$$
$$x\Big(\delta^+(S)\Big) - y_i \geq 0 \qquad \forall S \subset V - \{v_0\}, \text{ and } 2 \leq |S| \leq n-2,\ i \in S \tag{11}$$
$$x_a - M y_i \leq 0 \qquad \forall i \in V, a \in A \text{ with } he_a = i \tag{12}$$
$$x_a \text{ positive integer and } y_i \in \{0,1\} \qquad \forall a \in A \text{ and } \forall i \in V \tag{13}$$

where M is a large number representing an upper bound on the number of times an arc is used. In this formulation, constraints (8)–(11) have the same meaning as in formulation F1, and constraints (12) are used to express the relation between the two types of variables. As in [6], M is set to $|A| + 1$.

We now describe a new formulation for the CEARP, which we call F3. Let y_a be a binary variable that represents the use of arc a with service. The integer variable x_a denotes the number of times that arc a is used without service. Then the CEARP can be stated as:

F3: Minimize $$\sum_{a \in A} c_a (x_a + y_a) \tag{14}$$

subject to
$$x\Big(\delta^+(0)\Big) + y\Big(\delta^+(0)\Big) \geq 1 \tag{15}$$
$$x\Big(\delta^+(i)\Big) + y\Big(\delta^+(i)\Big) - x\Big(\delta^-(i)\Big) - y\Big(\delta^-(i)\Big) = 0 \qquad \forall i \in V \tag{16}$$
$$\sum_{a \in A} \lambda_{wa} y_a \geq 1 \qquad \forall w \in W \tag{17}$$
$$x\Big(\delta^+(S)\Big) + y\Big(\delta^+(S)\Big) - y_a \geq 0 \qquad \forall S \subset V - \{v_0\}, \text{ and } 2 \leq |S| \leq n-2,\ a \in A(S) \tag{18}$$
$$x_a \text{ positive integer and } y_a \in \{0,1\} \qquad \forall a \in A. \tag{19}$$

	F1	F2	F3
No. variables	$\|A\|$	$\|A\| + \|V\|$	$2.\|A\|$
No. constraints	$\|V\| + \|W\| + 1$	$\|V\| + \|A\| + \|W\| + 1$	$\|V\| + \|W\| + 1$
Big-M	Yes	Yes	No

Table 1: Comparison of the three formulations

The meaning of each constraint in F3 is as in F1. Constraints (15)–(18) imply constraints (2)–(5), respectively.

It is easy to see that the three formulations F1, F2, and F3 are equivalent. Table 1 gives a comparison of the three formulations. F1 has the smallest number of variables and constraints (we do not consider the connectivity constraints here because of their exponential number). The most important disadvantage of F1 is that it uses a large number M in the connectivity constraint. Because we can not find an efficient way to closely estimate M, an exact procedure to separate this constraint is useless. Experiments show that the performance of the branch-and-cut algorithm based on this formulation is poor compared to that of the other two formulations. F3 has the largest number of variables, but it also has some strong advantages. First, it does not contain the Big-M constraints that are known to weaken the linear relaxation and to decrease the performance of MILP models. Second, the set covering polytope with binary variables has received more attention than that with integer variables. Hence, the identification of violated constraints of type (17) in F3 seems to be more favorable than that for (4) in F1 or (10) in F2. Formulation F2 has fewer variables but more constraints than F3. Moreover, it contains Big-M constraints.

2.2. Valid equalities and inequalities

In this subsection, we introduce several valid equalities and inequalities for F3. Let $A' \subset A$ be a set of arcs that can not cover any customer. The following dominance relation is valid for the CEARP:

$$y_a = 0 \qquad \forall a \in A'. \tag{20}$$

Lemma 1. *Without loss of generality, we can assume that in an optimal solution $x_a > 0$ implies $y_a = 1 \; \forall \; a \notin A'$.*

Proof. Suppose that (x^*, y^*) is an optimal solution such that $x^*_{a'} > 0$ and $y^*_{a'} = 0$ for some arc $a' \notin A'$. Define $(x_a, y_a) = (x^*_a, y^*_a), \forall a \neq a'$ and $(x'_a, y'_a) =$

$(x^*_{a'} - 1, 1)$. It is easy to prove that this new solution is also optimal because it satisfies the constraints (15), (16), (17), (18) and has the same objective value (14). □

We can derive from Lemma 1 the following dominance property:

$$My_a \geq x_a \qquad \forall a \in A \setminus A' \tag{21}$$

where M is a large number representing an upper bound on the number of times an arc a is used. Its value can be taken as $|A|$.

Dominance constraints can also be derived, based on covering considerations. Let $(a_i, a_j) \in A \setminus A'$. Arc a_i is said to dominate arc a_j if for all $w_l \in W$, $\lambda_{il} \geq \lambda_{jl}$. The following constraints proposed in [7] are valid for the CEARP:

$$\begin{aligned} y_i + y_j \leq 1 \qquad & \forall a_i, a_j \in A \setminus A' \\ & \text{and if } a_i \text{ dominates } a_j, \text{ or conversely.} \end{aligned} \tag{22}$$

We also note that the constraints (21) and (22) can not be used at the same time because of contrary properties. The reason is that the former implies that if an arc that can cover a customer is traversed, then it is always used with service while the latter limits the use of arcs with service only when necessary.

All the valid inequalities of the set covering polytope $conv\{x : \sum b_a.y_a \geq 1, y_a \in \{0,1\}\}$ are valid for the CEARP. Balas and Ng [12] proposed the facets with coefficients in {0,1,2} and Sánchez-García et al. [13] introduced the more complex facets with coefficients in {0,1,2,3}. Here, we recall the first inequality that was used in [7]. Let S be a nonempty subset of W and define for each $a \in A$ the coefficient

$$\alpha_a^S = \begin{cases} 0 \ \texttt{if} \ \lambda_{al} = 0 \ \texttt{for all} \ w_l \in S, \\ 2 \ \texttt{if} \ \lambda_{al} = 1 \ \texttt{for all} \ w_l \in S, \\ 1 \ \texttt{otherwise}. \end{cases}$$

Then the following inequality is valid for the CEARP:

$$\sum_{a \in A} \alpha_a^S y_a \geq 2 \qquad \forall a \in A \setminus A' \text{ and } \forall S \subset W : |S| > 0. \tag{23}$$

The connectivity constraints (18) can be strengthened by adding information about the covering constraint. The following constraint is a stronger form of connectivity:

$$x\Big(\delta^+(S)\Big) + y\Big(\delta^+(S)\Big) \geq 1 \tag{24}$$

where $S \subset V$ and at least one customer of W is not covered by any arc in $A(V \setminus S)$.

2.3. Separation of cuts

We now present separation procedures for each class of valid inequalities described in the previous subsection. To simplify the presentation, we give the details only for F3. The separation of cuts for F2 is similar. Let (x^*, y^*) be a fractional solution to be separated. Let G^* be the weighted graph induced by (x^*, y^*) such that the capacity of each arc $C_a = x^*_a + y^*_a$.

The detection of constraints of type (20), (21), and (22) is straightforward. Although these constraints help to reduce the number of vertices in the branch-and-bound tree, constraints (21) contain the Big-M that decreases the performance of the branch-and-cut algorithm. Since M is determined by the number of arcs, the larger the instance, the more important the negative impact of M. Our tests show that when M can not be closely estimated, a branch-and-cut algorithm without Big-M constraints is normally more efficient than one with Big-M. This is why we use the constraints (20) and (22) in the general case where M can not be estimated effectively. The constraints (21) are only used if we can find the way to bound M strictly. Whenever these dominance constraints are used, they are directly included in the initial model because they are almost certainly violated.

For constraints (23), we tested $|S| = 3$, as in [7], to reduce the computational effort. The process was still time-consuming because of the large number of customers in the CEARP instances. However, we observe that the zero-half cut of CPLEX can generate this type of constraint. Our tests show that using the CPLEX cut is more effective and faster than directly applying (23).

We will present an example to show how a zero-half cut can generate the constraints (23). Suppose there is a graph with three arcs a_1, a_2, and a_3 serving three customers w_1, w_2, and w_3. These customers can be served by the arcs (a_2, a_3), (a_1, a_3), and (a_1, a_2), respectively. The covering constraints (17) for three customers are as follows: $y_2 + y_3 \geq 1, y_1 + y_3 \geq 1$, and $y_1 + y_2 \geq 1$.

By considering the sum of these three constraints, dividing the resulting constraint by 2, and rounding its right-hand side, we get the zero-half cut: $y_1 + y_2 + y_3 \geq 2$. This constraint has the same form as (23). Hence, the CPLEX zero-half cut can generate the constraints (23).

The connectivity constraints (18) can be separated by both heuristic and exact methods. In the heuristic method, we first compute the connected components $S_1, S_2, ..., S_q$ of the subgraph G^*. If the number of connected components is at least two (i.e., $q \geq 2$) then $x\Big(\delta^+(S_i)\Big) + y\Big(\delta^+(S_i)\Big) \geq y_a$, for each component i and each $a \in A(S_i)$, is a violated inequality. We can also separate these constraints exactly in polynomial time by applying the following algorithm:

- For each node $i \in V \setminus \{v_0\}$: Compute the maximum flow from i to depot v_0. Let $\delta^+(S_i)$, $S_i \subseteq V \setminus \{v_0\}$ be the min-cut and let f_i be the value of this flow.
- For each arc $a = (v_i, v_j) \in A$ and $x_a \neq 0$: If $f_i < x_a$, then a violated connectivity constraint has been found.

Let f_a be the maximum flow passing through arc $a = (v_i, v_j)$. To ensure connectivity from the depot to arc a, there must be at least a flow of x_a passing through a. Note that $f_i \geq f_a$. Then, if $f_i < x_a$, we have $f_a < x_a$. Hence, $f_i < x_a$ is a violated constraint. On the other hand, if $f_i \geq x_a$, we can always send a flow of at least x_a through a and connectivity is ensured. If the Edmonds-Karp algorithm, which runs in $\mathcal{O}(|V||A|^2)$ time, is used to solve the maximum flow problems, the running time of this procedure is $\mathcal{O}(|V|^2|A|^2)$.

Finally, constraints (24) can be generated by a heuristic similar to that proposed for (18). Compute the connected components $S_1, S_2, ..., S_q$ of the subgraph $G(x^*, y^*)$ induced by the arcs $a \in A$ with $x^* + y^* \geq \epsilon$, where $0 \leq \epsilon < 1$ is a given parameter. If the number of connected components is at least two (i.e., $q \geq 2$), and the arcs with two endpoints in $V \setminus S$ can not cover all the customers and $x^*\Big(\delta^+(S_i)\Big) + y^*\Big(\delta^+(S_i)\Big) < 1$ then $x\Big(\delta^+(S_i)\Big) + y\Big(\delta^+(S_i)\Big) \geq 1$ for each component $i = 1, ..., q$ is a violated inequality.

2.4. Upper bound for CEARP

In this subsection, we introduce a fast heuristic called UB1 that gives feasible solutions for the CEARP. This heuristic is used in the exact algorithm

to provide an initial upper bound. It is based on the algorithm proposed for the DRPP [2] and is as follows:

Step 1. Solve an integer program including the constraints (1), (3), (4), and (6) using CPLEX. The solution of this MIP is rapid even for the large instances in our tests.

Step 2. Construct a directed graph $G_R = (V_R, A_R)$ induced by the solution from Step 1, adding the depot if it is not already present. If G_R is connected then stop; the solution found in Step 1 is also a feasible solution for the CEARP. Otherwise, go to Step 3.

Step 3. Compute the connected components $C_1, C_2, ..., C_k$ of the graph G_R. Let K_i be the set of vertices corresponding to the connected component i. Build the undirected graph $\bar{G} = (N, E)$ with set of vertices $N = 1, ..., k$ and set of edges $E = \{(i, j), i, j \in N, i \neq j$. The corresponding edge costs are $c_{ij} = \min\{c(p, q) + c(q, p), p \in K_i, q \in K_j\}$ for each edge $(i, j) \in E$, where $c(p, q)$ is the length of the shortest path from vertex p in component i to vertex q in component j. Determine a minimum-cost spanning tree in $\bar{G}$. Let A_T be the set of arcs in the original graph corresponding to the edges in the tree. Add to $\bar{G}$ all the arcs of A_T.

2.5. Branch-and-cut algorithms

We first introduce a procedure that can reduce the number of customers. Given $w_l \in W$, let $Z(w_l)$ be the set of arcs that can cover w_l. For each pair of customers w_i and w_j, if $Z(w_i) \subseteq Z(w_j)$ then customer w_j can be ignored. This is because when w_i is served, w_j is covered at the same time. Note that the number of customers remaining is also the maximum number of arcs that must be activated for covering purposes. This procedure eliminates at least 50% of the customers in our tests.

We solve the CEARP exactly using a classic branch-and-cut algorithm. To simplify the description we describe the algorithm only for F3; the implementation is similar for F1 and F2. We solve a linear program containing the constraints (15), (16), (17), (20), (21) (or (22)), and constraints $0 \leq y_a \leq 1$. We then search for violated constraints of type (18) and (24), and the constraints detected are added to the current LP, which is then reoptimized. This process is repeated until all the constraints are satisfied. If there are fractional variables, we branch to generate two new subproblems. If all the variables are integer, we explore another subproblem.

Because the exact separation for (18) is quite time-consuming, after numerous tests, we decided to apply it only at the root node. More precisely,

at every node, we first find the strongly connected components of the graph created by the current variables. For each component, we check which customers are covered. If these customers can not be covered by the arcs outside the component, a constraint of type (24) is found. Otherwise, a constraint of type (18) is detected and added to the current program. At the root node, if this procedure fails to find violated constraints, we carry out the exact separation method. In other words, the exact separation of constraint (18) is used at the root node only if the heuristic fails. At the other nodes, only heuristic separation is applied. For F1, since we can not find any way to closely estimate M in constraints (5), the exact separation is useless for these constraints. Therefore, only the heuristic method is used even at the root node.

Our branch-and-cut algorithm is built around CPLEX 11.2 with the Callable Library. All CPLEX cuts except the zero-half cut are turned off. The parameter CPX–PARAM–ZEROHALFCUTS is set to 2 to generate zero-half cuts aggressively. All the other CPLEX parameters are set to their default values.

We tested several branching techniques, such as branching on the variables y before x as in [7] and branching on the variables x before y, but these do not outperform the CPLEX branching. Hence, we let CPLEX make the branching decisions.

In [6], a branch-and-cut algorithm based on F2 is developed to solve the GDRPP. The differences with our algorithm are that all the cuts of CPLEX 9.1 are turned on by default, the constraints (24) are not used, and the exact method to separate the connectivity constraints (11) is used at all nodes in the search tree. We also note that zero-half cuts were not implemented in CPLEX 9.1.

2.6. A MIP-based constructive algorithm

In this subsection, we propose an algorithm called UB2 that gives good solutions for the CEARP. In practice, arcs are usually traversed just a few times, and the number of traversals is much lower than the lowest provable value of M. This suggests that we can bound the large number M in the branch-and-cut algorithm by a small value to improve performance. To ensure that a solution exists, a Chinese postman problem (CPP) is first solved, and M is determined by the maximum number of times an arc is traversed in the CPP solution. We use only F2 and F3 to construct this algorithm. For

F3, the dominance constraint (21) is used and added directly to the initial model.

3. Computational experiments

In this section, we describe the CEARP instances and the computational evaluation of the proposed algorithms. Our algorithm is coded in C/C++ and is run on a 2.4 GHz CPU with 6 GB of RAM. The running time of the branch-and-cut algorithms is limited to 2 hours for each instance.

3.1. Data instances

We first use the CEARP instances of [5], which are random instances based on directed graphs. We now recall how to build these instances. Graphs that imitate real street networks are first generated randomly; this procedure is as follows:

- The coordinates of n vertices are randomly generated in a unit square. Then a heuristic is used to find the shortest tour passing through all the nodes exactly once. This tour is a Hamiltonian circuit, and it is used as a framework to construct the full graph. The resulting graph is therefore strongly connected.

- To imitate real networks, random arcs are added to the current tour to give a total of $m = nd$ arcs, where n is the number of vertices and d the ratio between the number of arcs and the number of nodes, in such a way that: (i) the arcs are not too long, and (ii) there is no intersection between any two arcs.

In the tests in [5], graphs with $n \in \{300, 400, 500\}$ vertices and a ratio $d \in \{1.5, 2, 2.5, 3\}$ are used. For each combination of n and d, five different graphs are generated. The cost of an arc (v_i, v_j) is the Euclidean distance between v_i and v_j multiplied by five to obtain an average arc length close to that seen in practice (from about 0.2 to 0.4 km).

Once the graphs have been built, the CEARP instances are generated by randomly positioning $q = mt$ customer nodes in the square containing the graph, where m is the number of arcs and t the ratio between the number of customers and the number of arcs, $t \in \{0.5, 1, 5, 10\}$. Thus, for each graph, four CEARP instances are created. The effective radius r is set to 150 m. The distance between the arcs and the customer vertices are computed by

the distance from the closest point of the arc to the customer. To ensure that a solution exists, we delete all the customers that can not be covered by any arc. We also examine the impact of increasing the radius parameter from 150 m to 200 m. To do this, we use the graph created with $r = 150$ m but change the read range to 200 m.

From the instances of [5], we choose the largest ones with 500 nodes and 1500 arcs (r=150 m), and 500 nodes and 1000 arcs (r=200 m) to test our algorithms. The instances are labeled ce-n-k-r-t-i, where n is the number of nodes, k is the number of arcs, r is the read range, t is the ratio between the number of customers and the number of arcs, and i (=1,...,5) is the instance number. For example, ce-500-1500-150-10-5 indicates the 5th instance with 500 nodes, 1500 arcs, 150 m of read range, and t =10.

We also use mixed graphs from the literature to generate instances for the CEARP. We choose the mixed graphs introduced in [14] for which the coordinates of the vertices are published. These are large graphs with a structure similar to that of real street networks. To transform the mixed graphs to directed graphs, we model each undirected edge by two arcs with the same cost. From these graphs, we select two, MB537 and MB547, which, after being transformed to directed graphs, have fewer than 1500 arcs. MB357 has 500 nodes, 364 edges, and 476 arcs and MB547 has 500 nodes, 351 edges, and 681 arcs.

For the mixed graphs, the procedure to generate the customers is the same, except that the read range r is determined by the average length of all the arcs in the graph. For each graph and each value of t, we generate five CEARP instances.

We also test our algorithms on 30 instances defined on undirected graphs. We use two sets of graphs taken from [15] with 15 general routing problem instances each, generated from the Albaida and Madrigueras graphs proposed in [16] by defining each edge as being required with probability $P = \{0.3, 0.5, 0.7\}$. The Albaida graph includes 116 nodes and 174 edges, and the Madrigueras graph has 196 nodes and 316 edges. From these graphs, we generate CEARP instances as follows:

- The number of clients is defined by the number of required edges in the graph; each client is covered by a required edge.
- Each client is covered by e additional required edges where e is a random number taking values from 1 to 5, so that each client is covered by at least 2 and at most 6 edges.

To solve the undirected instances as directed instances, we transform each edge into two arcs with the same cost. As in other arc routing problems defined on undirected graphs (see [17], for example), it is easy to prove that, for a given CEARP instance defined on an undirected graph, an optimal solution exists in which no edge is traversed more than twice. This allows us to fix the large number M to 2 in F2 and to 1 in F3. Therefore, we use the dominance constraints of type (21) for the undirected instances. Note that, in this case, all variables in F3 are now binary. For the directed and mixed instances where M cannot be estimated effectively, the constraints (22) are used.

3.2. Comparisons of lower bounds

The first set of results compares the lower bounds obtained at the root node of our branch-and-cut algorithms. These are based on the formulations F1–F3 and the formulation similar to that in [6] in which all the CPLEX cuts except the zero-half cuts are turned on. The formulations are tested on the directed-graph instances of [5]. The results are given in Table 2. This table presents, for each formulation F1, F2 and F3, the lower bound obtained at the root node (lb_i). The value lb_4 is the lower bound obtained by the branch-and-cut algorithm of [6]. The best results are in bold.

The results shown in Table 2 indicate that the lb_4 bounds are always worse than the lb_2 bounds. This proves the efficiency of the addition of zero-half cuts in CPLEX. The results also imply that the formulations F2 and F3 are more efficient than F1. It seems that F3 is slightly better than F2, since lb_3 is larger than lb_2 in 30 of the 40 instances.

3.3. Overall comparisons

Table 3 compares the five exact algorithms for the CEARP in terms of the number of instances successfully solved to optimality. Column F0 gives the results for the cutting-plane method introduced in [5], and columns F1, F2, and F3 give the results for our three branch-and-cut algorithms based on the corresponding formulations. Column F4 presents the number of successful instances for the algorithm proposed in [6] using CPLEX 11.2 instead of 9.1 which contains, among other improvements, the zero-half cuts. Again, the formulations are tested on the directed-graph instances of [5].

From Table 3, an interesting observation is that the branch-and-cut algorithms based on F1 and proposed in [6] are worse than the cutting-plane method introduced in [5]. The branch-and-cut algorithms based on F2 and

Instance	lb_1	lb_2	lb_3	lb_4
ce500-1500-150-0.5-1	97408.9	100806.7	**102217.6**	97120.2
ce500-1500-150-0.5-2	91174.4	**94106.2**	93793.5	90109.9
ce500-1500-150-0.5-3	105610.5	109639.3	**109641.2**	102412.4
ce500-1500-150-0.5-4	89407.1	**92052.3**	92023.0	86579.5
ce500-1500-150-0.5-5	97380.9	99276.9	**100212.6**	93786.0
ce500-1500-150-1-1	124365.7	126587.0	**127317.0**	122182.1
ce500-1500-150-1-2	119585.3	121820.3	**122007.2**	116832.0
ce500-1500-150-1-3	129284.2	**131544.9**	131222.7	126613.8
ce500-1500-150-1-4	109707.3	112258.5	**113558.1**	108049.6
ce500-1500-150-1-5	112212.1	**114481.8**	113600.8	106513.5
ce500-1500-150-5-1	158893.7	160143.6	**160772.9**	157372.1
ce500-1500-150-5-2	158452.9	159078.5	**159897.7**	157100.3
ce500-1500-150-5-3	174584.4	**176492.5**	176166.2	171283.5
ce500-1500-150-5-4	149310.1	150954.0	**150965.4**	147512.2
ce500-1500-150-5-5	159917.1	161005.9	**161236.8**	156457.5
ce500-1500-150-10-1	172904.7	173757.7	**174219.1**	170685.1
ce500-1500-150-10-2	171701.5	172117.5	**172691.4**	169647.9
ce500-1500-150-10-3	183425.9	184383.5	**184608.3**	180971.4
ce500-1500-150-10-4	160815.3	161200.2	**161279.4**	157174.1
ce500-1500-150-10-5	166463.3	166789.9	**167919.9**	163820.5
ce500-1000-200-0.5-1	68866.5	70697.9	**73487.9**	65020.4
ce500-1000-200-0.5-2	75673.2	79838.8	**82208.3**	74942.7
ce500-1000-200-0.5-3	75831.4	79291.9	**86063.8**	70341.0
ce500-1000-200-0.5-4	64885.7	72082.7	**73116.8**	68179.2
ce500-1000-200-0.5-5	78765.6	**83502.2**	83082.8	79526.7
ce500-1000-200-1-1	75801.9	79880.2	**80334.8**	71999.5
ce500-1000-200-1-2	83930.2	85511.7	**88301.9**	75334.9
ce500-1000-200-1-3	90527.1	94697.9	**95175.6**	90869.7
ce500-1000-200-1-4	73571.1	77859.5	**79749.0**	75844.6
ce500-1000-200-1-5	87119.4	**91569.9**	90790.5	88867.4
ce500-1000-200-5-1	94893.6	98107.2	**98547.0**	93816.3
ce500-1000-200-5-2	105008.9	108094.2	**108484.0**	104636.4
ce500-1000-200-5-3	110631.7	**113553.4**	112952.4	112829.9
ce500-1000-200-5-4	98142.1	99483.7	**103466.7**	97615.6
ce500-1000-200-5-5	108961.5	111696.9	**111932.5**	110569.1
ce500-1000-200-10-1	108402.9	**110222.7**	109062.8	107673.1
ce500-1000-200-10-2	108827.2	**112565.9**	112129.0	110069.1
ce500-1000-200-10-3	123741.6	125644.3	**125647.7**	124849.3
ce500-1000-200-10-4	111134.0	112184.6	**115022.7**	111502.0
ce500-1000-200-10-5	124283.3	126852.4	**127171.7**	124776.1

Table 2: Comparison of lower bounds of the four algorithms

Data	F0	F1	F2	F3	F4
ce500-1500-0.5	0	0	0	0	0
ce500-1500-1	1	0	2	2	0
ce500-1500-5	4	3	5	5	2
ce500-1500-10	5	5	5	5	4
ce500-1000-0.5	2	0	1	3	1
ce500-1000-1	3	2	4	4	2
ce500-1000-5	5	3	5	5	4
ce500-1000-10	5	5	5	5	5

Table 3: Number of successful instances

F3 outperform the others. For $r = 150\,\mathrm{m}$, it seems that F2 and F3 are equivalent. For $r = 200\,\mathrm{m}$, F3 is better because it can solve 2 instances with $t = 0.5$ that F2 can not. Therefore, in the next subsection, we compare only F2 and F3 on other criteria and other instances.

3.4. Detailed comparisons of directed-graph instances

This subsection provides the results for the branch-and-cut algorithms using F2 and F3. For each formulation, Table 4 shows the time required (time), the gap (gap), and the solution value (result), and Table 5 presents the number of dominance constraints of type (22) (*domi*), the number of zero-half cuts (*zero*), the number of connectivity cuts (*user*), and the number of vertices (*bb*) in the branch-and-bound tree. The gap for F3 is better than that for F2. F3 also finds better solutions than F2. We believe this is because F2 contains Big-M constraints that lead to numerical difficulties because of the many generated connectivity constraints (the *user* column in Table 5). These increase the size of the model, and so CPLEX needs more time to process each node. Therefore, there are less chances to find good solutions. In contrast, the branch-and-bound tree of F3 is much larger. This is because we use the constraints (22) instead of the constraints (21) to reduce the size of the search tree and it seems that the constraints (21) are more efficient than the constraints (22).

3.5. Comparisons of undirected- and mixed-graph instances

Table 6 gives the results for the two formulations on undirected-graph instances. F3 can solve the instance MADR-7-3 that F2 can not. It is also faster on 22 of 30 instances. Therefore, on undirected instances F3 outperforms F2. Note that in this case, all the variables of F3 are binary and this is probably why F3 is better than F2.

	F2			F3		
Data	time	gap	result	time	gap	result
ce500-1500-150-0.5-1	7202.29	4.21	106474.2	7202.77	1.76	105231.4
ce500-1500-150-0.5-2	7202.72	3.34	97900.6	7203.01	2.13	97435.3
ce500-1500-150-0.5-3	7202.46	0.82	112363.3	7202.91	0.39	112129.6
ce500-1500-150-0.5-4	7202.45	2.23	95657.7	7202.98	2.53	95778.1
ce500-1500-150-0.5-5	7202.51	2.35	102603.2	7202.91	0.98	102323.9
ce500-1500-150-1-1	7206.93	1.25	129790.7	7207.52	0.40	129659.7
ce500-1500-150-1-2	505.35	0	123123.0	278.86	0	123123.0
ce500-1500-150-1-3	121.73	0	133418.3	598.91	0	133418.3
ce500-1500-150-1-4	7206.53	2.11	116531.6	7207.07	1.04	116058.8
ce500-1500-150-1-5	7206.48	2.21	117967.2	7206.98	1.55	117003.4
ce500-1500-150-5-1	452.86	0	162097.8	278.01	0	162097.8
ce500-1500-150-5-2	128.71	0	160792.7	101.33	0	160792.7
ce500-1500-150-5-3	166.73	0	177242.4	122.72	0	177242.4
ce500-1500-150-5-4	64.19	0	151852.9	78.69	0	151852.9
ce500-1500-150-5-5	147.09	0	161833.4	191.63	0	161833.4
ce500-1500-150-10-1	152.53	0	174504.1	174.82	0	174504.1
ce500-1500-150-10-2	212.81	0	173404.5	190.57	0	173404.5
ce500-1500-150-10-3	197.64	0	185330.8	201.41	0	185330.8
ce500-1500-150-10-4	201.64	0	162071.7	222.80	0	162071.7
ce500-1500-150-10-5	945.25	0	168734.3	240.02	0	168734.3
ce500-1000-200-0.5-1	7200.69	1.59	76033.0	4275.83	0	76033.0
ce500-1000-200-0.5-2	7200.78	1.22	84311.6	385.83	0	84237.2
ce500-1000-200-0.5-3	7200.74	12.82	95790.2	7200.95	0.80	89653.5
ce500-1000-200-0.5-4	7200.71	2.51	76226.2	7201.04	1.52	76084.4
ce500-1000-200-0.5-5	137.05	0	85097.0	1714.54	0	85097.0
ce500-1000-200-1-1	461.58	0	82687.1	4440.20	0	82687.1
ce500-1000-200-1-2	140.30	0	89896.5	257.57	0	89896.5
ce500-1000-200-1-3	6209.50	0	98051.8	279.85	0	98051.8
ce500-1000-200-1-4	7202.02	1.95	82682.3	7202.20	0.41	82344.2
ce500-1000-200-1-5	38.03	0	91915.6	59.10	0	91915.6
ce500-1000-200-5-1	130.37	0	100395.3	1233.24	0	100395.3
ce500-1000-200-5-2	36.33	0	109318.5	92.57	0	109318.5
ce500-1000-200-5-3	121.71	0	114362.3	121.69	0	114362.3
ce500-1000-200-5-4	57.89	0	103790.9	75.01	0	103970.9
ce500-1000-200-5-5	19.58	0	112625.0	52.97	0	112625.0
ce500-1000-200-10-1	67.97	0	110527.9	119.54	0	110527.9
ce500-1000-200-10-2	57.21	0	113694.1	83.24	0	113694.1
ce500-1000-200-10-3	82.84	0	126659.9	95.27	0	123733.9
ce500-1000-200-10-4	69.35	0	115281.4	54.81	0	115281.4
ce500-1000-200-10-5	44.56	0	128163.2	58.86	0	128163.2

Table 4: Comparison of performance of F2 and F3

	F2			F3			
Data	*zero*	*user*	*bb*	*domi*	*zero*	*user*	*bb*
ce500-1500-150-0.5-1	259	19913	5817	3293	1910	9334	61903
ce500-1500-150-0.5-2	263	20036	6236	3575	2374	3701	113037
ce500-1500-150-0.5-3	390	9797	16080	2998	2785	1818	169909
ce500-1500-150-0.5-4	592	7189	30983	3469	2658	4539	71536
ce500-1500-150-0.5-5	349	17751	12756	3333	2426	4453	66482
ce500-1500-150-1-1	269	16305	11690	2797	2567	1540	264117
ce500-1500-150-1-2	686	2015	16271	2644	1569	361	20852
ce500-1500-150-1-3	427	1229	3969	2900	2053	1300	40877
ce500-1500-150-1-4	418	7696	22392	2791	2101	2778	125514
ce500-1500-150-1-5	262	15410	12192	3044	2029	6288	48432
ce500-1500-150-5-1	355	3663	3119	2243	1140	264	33910
ce500-1500-150-5-2	340	535	1270	2293	664	177	1561
ce500-1500-150-5-3	246	2040	343	2444	467	328	716
ce500-1500-150-5-4	231	158	136	2159	529	124	1047
ce500-1500-150-5-5	366	224	582	2226	984	149	4817
ce500-1500-150-10-1	242	164	174	2152	400	46	144
ce500-1500-150-10-2	289	1013	1286	2425	667	135	1158
ce500-1500-150-10-3	244	512	141	2259	376	448	487
ce500-1500-150-10-4	301	211	292	2396	806	92	2667
ce500-1500-150-10-5	639	6493	5309	2152	948	129	2741
ce500-1000-200-0.5-1	200	17037	11274	2110	3179	2845	180230
ce500-1000-200-0.5-2	160	27332	8960	1972	1618	1064	26006
ce500-1000-200-0.5-3	116	37116	1143	1942	4397	2381	194081
ce500-1000-200-0.5-4	196	18965	21752	1989	1572	8734	64879
ce500-1000-200-0.5-5	218	3417	1146	1994	3372	2458	51554
ce500-1000-200-1-1	272	6291	4822	1666	2106	6997	104788
ce500-1000-200-1-2	338	2709	3206	1590	1668	1232	10894
ce500-1000-200-1-3	148	29035	2314	1520	1613	738	18844
ce500-1000-200-1-4	230	19286	17746	1444	681	8147	79627
ce500-1000-200-1-5	160	945	70	1456	636	495	3987
ce500-1000-200-5-1	214	2670	785	1146	2354	2546	64371
ce500-1000-200-5-2	182	221	125	1245	527	162	1329
ce500-1000-200-5-3	164	2870	285	1024	652	1656	3312
ce500-1000-200-5-4	161	1376	176	1070	385	137	266
ce500-1000-200-5-5	142	156	27	1208	339	264	194
ce500-1000-200-10-1	162	298	46	1020	568	257	2052
ce500-1000-200-10-2	273	810	194	1306	812	116	5612
ce500-1000-200-10-3	146	979	305	1067	421	733	1549
ce500-1000-200-10-4	133	961	79	1104	224	21	18
ce500-1000-200-10-5	134	192	27	1191	449	107	468

Table 5: Details of branch-and-cut algorithms for F2 and F3

	F2					F3				
Data	lb1	gap	bb	time	result	lb1	gap	bb	time	result
ALBA-3-1	2391.8	0	236	5.34	2511	2319.2	0	209	2.08	2511
ALBA-3-2	2124.7	0	119	4.14	2324	2011.2	0	74	2.15	2324
ALBA-3-3	2012.8	0	74	1.88	2155	2070.7	0	30	1.32	2155
ALBA-3-4	2363.7	0	959	53.42	3074	2621.0	0	334	13.39	3074
ALBA-3-5	2244.0	0	71	3.63	2440	2355.1	0	21	2.04	2440
ALBA-5-1	2845.0	0	703	28.15	3125	2712.3	0	1105	24.26	3125
ALBA-5-2	2468.3	0	637	24.70	2926	2480.5	0	1655	35.78	2926
ALBA-5-3	3013.8	0	279	9.93	3170	2961.5	0	446	13.12	3170
ALBA-5-4	2447.8	0	298	8.78	2584	2405.8	0	433	7.77	2584
ALBA-5-5	2472.4	0	477	24.66	2642	2535.6	0	112	3.22	2642
ALBA-7-1	2989.7	0	495	21.33	3397	2879.4	0	528	13.64	3397
ALBA-7-2	3200.1	0	793	29.71	3558	3192.1	0	740	21.77	3558
ALBA-7-3	3234.6	0	1939	68.88	3647	3125.3	0	1913	64.31	3647
ALBA-7-4	3255.8	0	365	15.20	3461	3264.4	0	789	24.63	3461
ALBA-7-5	2540.8	0	401	23.27	2821	2642.5	0	249	8.81	2821
MADR-3-1	2511.2	0	11215	2027.39	2925	2658.0	0	4757	137.62	2925
MADR-3-2	3103.3	0	10653	1972.94	3665	3197.5	0	28755	2731.95	3665
MADR-3-3	2580.7	0	2556	366.50	3045	2714.9	0	2516	112.25	3045
MADR-3-4	2829.3	0	8838	2463.51	3295	2936.4	0	11386	794.31	3295
MADR-3-5	2704.7	0	3603	552.06	3165	2760.4	0	7960	1303.2	3165
MADR-5-1	3479.6	0	7116	1398.53	3945	3568.7	0	4261	593.94	3945
MADR-5-2	4234.9	0	3124	280.35	4570	4239.4	0	6871	213.86	4570
MADR-5-3	3850.6	0	17362	4067.07	4505	3810.2	0	4395	480.13	4505
MADR-5-4	3780.9	0	1526	184.09	4020	3840.0	0	2605	180.52	4120
MADR-5-5	3583.5	0	7894	1156.25	4010	3695.5	0	6199	738.96	4010
MADR-7-1	4287.3	0	5041	850.95	4645	4329.4	0	14985	2512.30	4645
MADR-7-2	4363.8	0	7424	1499.57	4650	4350.3	0	4825	477.29	4650
MADR-7-3	4120.9	1.05	33074	7200.08	4620	4234.8	0	16184	2128.87	4620
MADR-7-4	4190.7	2.27	27065	7200.07	4655	4186.7	2.48	39664	7200.06	4645
MADR-7-5	4370.1	0	12706	2958.65	4735	4338.2	0	23533	4535.04	4735

Table 6: Comparison of performance of F2 and F3 on undirected instances

	F2					F3				
Data	succ	gap	bb	time	result	succ	gap	bb	time	result
MB0537-0.5	4	1.75	10330.2	3379.42	17592.0	0	1.07	38679.6	7200.72	17460.4
MB0537-1	3	0.59	10425.2	4707.06	18646.8	0	1.33	63897.2	7201.54	18671.4
MB0537-5	5	0	3242.0	185.89	21712.2	3	0.19	138095.6	3937.83	21712.2
MB0537-10	5	0	1040.2	74.01	22666.2	5	0	163733.6	2418.27	22666.2
MB0547-0.5	0	9.28	5450.2	7200.63	15559.8	0	6.95	19457.4	7201.09	15489.2
MB0547-1	0	3.38	14069	7201.74	17155.0	0	3.48	42625.4	7202.20	17216.2
MB0547-5	5	0	4428.6	470.27	21343.2	4	0.13	27233.8	1639.97	21343.2
MB0547-10	5	0	1461.6	79.97	22404.0	5	0	31992.4	756.16	22404.0

Table 7: Comparison of performance of F2 and F3 on mixed-graph instances

Table 7 gives the results for mixed-graph instances. All the results are averages over five instances. F2 can solve more instances than F3 but once again its gap is, in some cases, poor.

3.6. Results for upper bounds

Tables 8 and 9 give the results for algorithms UB1 and UB2. In these tables, M_{CPP} is the value of M calculated by solving the CPP problems. The names of instances in bold indicate that these instances were proved optimal by the branch-and-cut algorithms. The BnC Time column gives the running time in seconds of the branch-and-cut algorithm based on F2. The Gap column displays the gap in percent of upper bounds to the best solutions found by the two branch-and-cut algorithms in the previous subsection. The negative results imply that the upper-bound algorithms found better solutions than the branch-and-cut algorithms. From the results of the exact algorithms, we observe that an arc is rarely crossed more than 5 times in the solution. Therefore, we also test the MIP-based constructive algorithms with $M = 5$. Note that this algorithm is not competitive on undirected-graph instances because in these cases we can bound the large number M efficiently. When M is bounded more strictly, F2 is much better than F3, so we do not present the results of the algorithm for F3.

The results for directed- and mixed-graph instances show the good performance of the algorithm UB2. It not only finds better solutions but is also faster in almost all the instances that the branch-and-cut algorithms can not solve exactly. The performance of algorithm with $M = 5$ is slightly better than with $M = M_{CPP}$ but bounding M too strictly can make the problem infeasible, two instances ce500-1500-5-3-150 and ce500-1500-10-3-150 for example. The quality of UB1 is poor, especially on the directed-graph instances

where $r = 200\,\text{m}$ as well as on the mixed-graph instances. It only works better on the easy instances with $r = 150\,\text{m}$ and $t = 0.5, 1$. However, it is much faster than UB2.

4. Conclusion

We have proposed a new formulation for the CEARP. In contrast to two formulations in the literature, this formulation has an important advantage: it does not require Big-M constraints. Branch-and-cut algorithms have been developed for three formulations, and the three formulations have been compared. The results show that the branch-and-cut algorithms based on our new formulation and on the formulation of [6] outperform the other algorithms considered. In comparison with the formulation of [6], our new formulation is better on directed- and undirected-graph instances but worse on mixed-graph instances. Normally, our new formulation gives a better gap, but an enormous branch-and-bound tree is the price to pay. We also propose a MIP-based constructive heuristic for the CEARP on directed- and mixed-graph instances based on the formulation of [6] in which the large number M is bounded more strictly.

Data	UB1		UB2-F2, M=5		UB2-F2, M=M_{CPP}			Bnc
	Time	Gap	Time	Gap	M_{CPP}	Time	Gap	Time
ce500-1500-150-0.5-1	6.93	14.71	3819.53	-0.32	11	1128.79	-0.32	7202.29
ce500-1500-150-0.5-2	7.49	16.97	473.07	-0.69	12	1084.34	-0.69	7202.72
ce500-1500-150-0.5-3	7.80	15.81	141.14	-0.02	15	171.44	-0.02	7202.46
ce500-1500-150-0.5-4	8.28	13.30	1337.99	-0.80	12	7202.44	-0.79	7202.45
ce500-1500-150-0.5-5	7.30	13.97	1699.12	-0.33	10	7202.37	-0.33	7202.51
ce500-1500-150-1-1	10.97	10.02	143.32	-0.07	11	129.40	-0.07	7206.93
ce500-1500-150-1-2	11.25	6.24	83.17	0	12	90.92	0	505.35
ce500-1500-150-1-3	11.50	6.39	153.90	0	15	121.17	0	121.73
ce500-1500-150-1-4	11.60	12.62	480.28	-0.10	12	746.87	-0.10	7206.53
ce500-1500-150-1-5	11.19	11.38	7206.52	-0.12	10	7206.26	0.03	7206.48
ce500-1500-150-5-1	57.61	2.94	140.66	0	11	117.08	0	452.86
ce500-1500-150-5-2	54.70	2.41	133.76	0	12	132.17	0	128.71
ce500-1500-150-5-3	54.15	3.58			15	147.08	0	166.73
ce500-1500-150-5-4	51.24	3.62	93.06	0	12	79.82	0	64.19
ce500-1500-150-5-5	53.93	3.04	147.37	0	10	166.00	0	147.09
ce500-1500-150-10-1	119.36	1.75	143.05	0	11	152.90	0	152.53
ce500-1500-150-10-2	107.62	2.26	164.46	0	12	175.16	0	212.81
ce500-1500-150-10-3	118.05	2.57			15	178.73	0	197.64
ce500-1500-150-10-4	101.88	2.78	175.58	0	12	178.49	0	201.64
ce500-1500-150-10-5	115.54	2.91	250.78	0	10	218.16	0	945.25
ce500-1000-200-0.5-1	4.58	30.25	88.65	0	10	54.29	0	7200.69
ce500-1000-200-0.5-2	4.69	42.00	66.63	0	14	161.62	0	7200.78
ce500-1000-200-0.5-3	4.62	33.06	96.97	0	12	354.75	0	7200.74
ce500-1000-200-0.5-4	4.57	41.92	287.06	-0.17	12	252.59	-0.17	7200.71
ce500-1000-200-0.5-5	4.68	39.59	89.17	0	11	81.70	0	137.05
ce500-1000-200-1-1	5.78	24.22	78.95	0	10	106.86	0	461.58
ce500-1000-200-1-2	5.87	35.84	73.87	0	14	182.10	0	140.30
ce500-1000-200-1-3	6.07	23.36	180.66	0	12	81.72	0	6209.50
ce500-1000-200-1-4	5.94	30.50	112.85	0	12	358.85	0	7202.02
ce500-1000-200-1-5	5.99	28.21	23.47	0	11	15.51	0	38.03
ce500-1000-200-5-1	19.57	30.18	60.29	0	10	97.54	0	130.37
ce500-1000-200-5-2	17.98	10.98	74.72	0	14	81.14	0	36.33
ce500-1000-200-5-3	21.04	10.41	58.55	0	12	71.51	0	121.71
ce500-1000-200-5-4	21.14	18.55	34.15	0	12	42.40	0	57.89
ce500-1000-200-5-5	21.19	18.32	24.90	0	11	28.32	0	19.58
ce500-1000-200-10-1	39.85	9.79	67.88	0	10	95.28	0	67.97
ce500-1000-200-10-2	35.85	6.63	49.28	0	14	55.72	0	57.21
ce500-1000-200-10-3	42.70	4.45	80.61	0	12	86.33	0	82.84
ce500-1000-200-10-4	42.54	11.91	49.51	0	12	46.20	0	69.35
ce500-1000-200-10-5	43.33	12.98	62.41	0	11	57.18	0	444.56

Table 8: Upper bounds based on F2 for directed-graph instances

	UB1		UB2-F2, M=5		UB2-F2, M=M_{CPP}			BnC
Data	Time	Gap	Time	Gap	M_{CPP}	Time	Gap	Time
MB0537-0.5-1	4.77	34.40	345.39	0	7	278.38	0	992.57
MB0537-0.5-2	4.77	39.41	382.67	0	7	397.07	0	4491.65
MB0537-0.5-3	5.04	36.08	2726.26	-0.51	7	981.18	-0.51	7200.69
MB0537-0.5-4	4.73	37.74	146.45	0	7	212.68	0	690.54
MB0537-0.5-5	5.09	31.65	478.33	0	7	214.05	0	3521.67
MB0537-1-1	5.79	33.56	276.76	0	7	155.04	0	1382.62
MB0537-1-2	5.79	33.60	108.31	0	7	127.44	0	3629.55
MB0537-1-3	5.78	33.08	393.87	-0.45	7	287.98	-0.45	7201.13
MB0537-1-4	5.53	38.33	723.45	0	7	3143.52	0	7201.26
MB0537-1-5	5.88	34.87	2471.76	0	7	617.78	0	4120.76
MB0537-5-1	13.40	17.44	274.46	0	7	283.69	0	593.70
MB0537-5-2	13.08	15.59	41.81	0	7	50.01	0	60.67
MB0537-5-3	13.78	18.12	92.07	0	7	96.55	0	138.32
MB0537-5-4	14.10	17.96	63.01	0	7	107.87	0	93.19
MB0537-5-5	13.66	15.07	64.30	0	7	57.08	0	43.58
MB0537-10-1	23.27	13.45	69.48	0	7	72.61	0	73.15
MB0537-10-2	25.68	14.77	67.03	0	7	74.79	0	75.52
MB0537-10-3	25.64	18.29	185.10	0	7	144.82	0	114.87
MB0537-10-4	25.22	11.42	49.43	0	7	30.24	0	30.61
MB0537-10-5	24.12	14.85	40.86	0	7	46.35	0	75.92
MB0547-0.5-1	5.60	44.89	7200.58	-4.03	14	7200.88	-1.40	7200.65
MB0547-0.5-2	5.23	41.12	1460.73	-1.20	14	2645.67	-1.20	7200.60
MB0547-0.5-3	5.22	45.92	7200.57	-1.06	14	2215.78	-1.33	7200.59
MB0547-0.5-4	5.31	39.19	7200.69	-1.34	14	7201.31	-0.65	7200.62
MB0547-0.5-5	5.21	40.64	7200.58	-1.65	14	7200.61	-1.63	7200.68
MB0547-1-1	6.25	36.72	66.54	-0.17	14	233.73	-0.17	7201.69
MB0547-1-2	6.27	33.26	383.36	-0.31	14	1211.10	-0.31	7201.86
MB0547-1-3	6.54	26.39	7201.62	-1.06	14	7201.70	-1.09	7201.60
MB0547-1-4	6.20	32.59	954.23	-0.21	14	969.81	-0.21	7201.87
MB0547-1-5	6.21	33.52	7201.62	-0.69	14	7201.59	-0.66	7201.67
MB0547-5-1	17.35	29.24	17.54	0	14	24.49	0	46.89
MB0547-5-2	17.62	40.39	18.28	0	14	43.18	0	32.89
MB0547-5-3	18.28	139.02	18.05	0	14	67.34	0	2158.19
MB0547-5-4	17.27	14.59	33.64	0	14	50.53	0	40.31
MB0547-5-5	17.46	16.91	64.69	0	14	52.02	0	73.08
MB0547-10-1	22.94	44.51	17.99	0	14	48.91	0	98.00
MB0547-10-2	35.85	14.71	60.13	0	14	67.20	0	54.33
MB0547-10-3	32.65	12.50	83.55	0	14	105.39	0	138.32
MB0547-10-4	31.59	11.56	69.47	0	14	64.96	0	47.86
MB0547-10-5	31.88	14.39	55.85	0	14	67.30	0	61.35

Table 9: Upper bounds based on F2 for mixed-graph instances

References

[1] N. Christofides, V. Campos, A. Corberán, E. Mota, An algorithm for the rural postman problem on a directed graph, Mathematical Programming Studies 26 (1986) 155–166.

[2] M. O. Ball, M. J. Magazine, Sequencing of insertions in printed circuit board assembly, Operations Research 36 (2) (1988) 192–201.

[3] V. Campos, J. V. Savall, A computational study of several heuristics for the directed rural postman problem, Computational Optimization and Applications 4 (1) (1993) 67–77.

[4] B. Golden, S. Raghavan, E. Wasil, Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network, in: The Vehicle Routing Problem: Lastest Avances and New Challenges, Springer, 2008, pp. 487–501.

[5] M.-H. Ha, N. Bostel, A. Langevin, L.-M. Rouseau, An exact algorithm for close enough traveling salesman problem, in: C.-J. Luz, F. Valente (Eds.), Proceedings of the 1st International Conference on Operations Research and Enterprise Systems, Vilamoura, Algarve, Portugal, 2012, pp. 233–238.

[6] M. Drexl, On some generalized routing problems, Ph.d. dissertation, Rheinisch-Westfalishe Technische Hochschule Aachen (2003).

[7] M. Gendreau, G. Laporte, F. Semet, The covering tour problem, Operations Research 45 (1997) 568–576.

[8] R. Baldacci, M. A. Boschetti, V. Maniezzo, M. Zamboni, Scatter search methods for covering tour problem, in: R. Sharda, S. Voss, C. Rego, B. Alidaee (Eds.), Metaheuristic optimization via memory and evolution, Springer Verlag, 2005, pp. 55–91.

[9] M. Hachicha, M. J. Hodgson, G. Laporte, F. Semet, Heuristics for the multi-vehicle covering tour problem, Computers and Operations Research 27 (1) (2000) 29–42.

[10] D. Gulczynski, J. Heath, C. Price, The close enough traveling salesman problem: A discussion of several heuristics, in: F. Alt, M. Fu, B. Golden

(Eds.), Perspectives in Operations Research: Papers in Honor of Saul Gass 80th Birthday, Springer Verlag, 2006, pp. 271–283.

[11] J. Dong, N. Yang, M. Chen, Heuristic approaches for a tsp variant: The automatic meter reading shortest tour problem, in: E. Baker, A. Joseph, A. Medrotra, M. Trick (Eds.), Extending the Horizons: Advances in Computing, Optimisation, and Decision Technologies, Springer Verlag, 2007, pp. 145–163.

[12] E. Balas, S. M. Ng, On the set covering polytope: All the facets with coefficients in {0, 1, 2}, Mathematical Programming 43 (1–3) (1986) 57–69.

[13] M. Sánchez-García, M. I. Sobrón, B. Vitoriano, On the set covering polytope: Facets with coefficients in {0, 1, 2, 3}, Annals of Operations Research 81 (1998) 343–356.

[14] A. Corberán, E. Mota, J. M. Sanchis, A comparison of two different formulations for arc routing problems on mixed graphs, Computers and Operations Research 33 (12) (2006) 3384–3402.

[15] A. Corberán, A. N. Letchford, J. M. Sanchis, A cutting plane algorithm for the general routing problem, Mathematical Programming 90 (2001) 291–316.

[16] E. Benavent, A. Carrotta, A. Corberán, J. M. Sanchis, D. Vigo, Lower bounds and heuristics for the windy rural postman problem, European Journal of Operations Research (2) (2007) 855–869.

[17] G. Ghiani, G. Laporte, A branch-and-cut algorithm for the undirected rural postman problem, Mathematical Programming 87 (3) (2000) 467–481.

4

Problème de tournées couvrantes multi-véhicules

Ce chapitre présente la résolution d'un problème qui est une généralisation du CVRP et défini comme suit. Soit $G = (V \cup W, E_1 \cup E_2)$ un graphe non orienté, où $V \cup W$ est l'ensemble de nœuds et $E_1 \cup E_2$ est l'ensemble d'arêtes. $V = \{v_0, ..., v_{n-1}\}$ est l'ensemble de n nœuds qui peuvent être visités et $W = \{w_1, w_2, ..., w_l\}$ est l'ensemble des nœuds qui doivent être couverts. Dénotons $T = \{v_0, ..., v_{t-1}\}$, un sous-ensemble de V, l'ensemble de nœuds qui doivent être visités. v_0 est le nœud de dépôt ; m véhicules identiques sont localisés au dépôt. Un coût c_{ij} (longueur ou temps de parcours) est associé avec chaque arête de $E_1 = \{v_i, v_j : v_i, v_j \in V, i < j\}$ et une distance d_{ij} est associée avec chaque arête de $E_2 = \{v_i, v_j : v_i \in V \setminus T, v_j \in W\}$. Le mCTP consiste à déterminer m routes pour m véhicules telles que le coût total soit minimisé et

- Chaque route commence et se termine au dépôt ;
- Chaque nœud de T soit visité exactement une fois tandis que chaque nœud de $V \setminus T$ soit visité au plus une fois ;
- Chaque nœud j de W soit couvert par des routes, c'est-à-dire, se situe à une distance maximale r d'au moins un nœud de $V \setminus T$ qui est visité, où r est le rayon de couverture ;
- Le nombre de nœuds sur chaque route (sauf le dépôt) soit plus petit qu'une valeur fixe p ;
- La longueur de chaque route ne dépasse pas une limite fixe q.

Le mCTP est évidement NP-difficile car il se réduit à un CVRP avec demandes unitaires si $T = V$ ou à un CTP si les contraintes de capacité sont relaxées. Dans ce chapitre, nous résolvons un cas particulier du mCTP dans lequel la contrainte sur la longueur est relaxée, c'est-à-dire $q = +\infty$. Nos contributions sont que : nous présentons une nouvelle formulation pour le mCTP et proposons une méthode exacte pour ce problème, ainsi qu'une métaheuristique. Des évaluations numériques montrent que notre approche exacte est meilleure que la méthode de génération de colonnes proposée dans Jozefowiez [35], et notre métaheuristique donne des solutions de bonne qualité pour les instances testées.

Le chapitre est organisé comme suit : nous introduisons successivement une nouvelle formulation, des méthodes de résolution ainsi que des évaluations numériques pour le problème. Ensuite nous concluons nos travaux. Enfin nous terminons ce chapitre par présenter notre article qui a été publié dans la revue *European Journal of Operational Research*.

4.1 Nouvelle formulation pour le mCTP

Dans cette section, nous décrivons une formulation de programmation en nombres entiers pour le mCTP. Le graphe original G est d'abord étendu à $\overline{G} = (\overline{V} \cup W, \bar{E}_1 \cup E_2)$ en ajoutant un nouveau nœud v_n, qui est une copie du dépôt v_0. Nous avons $\overline{V} = V \cup \{v_n\}$, $V' = \overline{V} \setminus \{v_0, v_n\}$, $\overline{E} = E_1 \cup \{(v_i, v_n), v_i \in V'\}$, et $c_{in} = c_{0i}\ \forall\ v_i \in V'$.

Cette formulation demande deux variables de flux, f_{ij} et f_{ji}, pour représenter une arête d'une solution réalisable du mCTP sur laquelle le véhicule transporte une charge de p unités. Lorsqu'un véhicule parcourt de v_i à v_j, le flux f_{ij} représente le nombre de nœuds qui ont été déjà visités et le flux f_{ji} représente le nombre de nœuds que nous pouvons visiter encore (c'est-à-dire, f_{ji} = p - f_{ij}).

Soit x_{ij} la variable binaire égale à 1 si l'arête $\{v_i, v_j\}$ est utilisée dans la solution et 0 sinon. Dénotons y_i la variable binaire qui indique la présence du nœud v_i dans la solution. Nous définissons les coefficients binaires λ_{il} égaux à 1 si et seulement si $w_l \in W$ peut être couvert par $v_i \in V \setminus T$. Nous considérons la version dans laquelle le nombre de véhicules est variable. Alors le mCTP peut être décrit comme suit :

$$\text{Minimiser} \quad \sum_{\{v_i,v_j\}\in\overline{E}} c_{ij}x_{ij} \tag{4.1}$$

$$\text{sous les contraintes} \quad \sum_{v_i\in V\setminus T} \lambda_{il}y_i \geq 1 \qquad \forall w_l \in W \tag{4.2}$$

$$\sum_{v_i\in\overline{V},i<k} x_{ik} + \sum_{v_j\in\overline{V},j>k} x_{kj} = 2y_k \qquad \forall v_k \in V' \tag{4.3}$$

$$\sum_{v_j\in\overline{V}} (f_{ji} - f_{ij}) = 2y_i \qquad \forall v_k \in V' \tag{4.4}$$

$$\sum_{v_j\in V'} f_{0j} = \sum_{v_i\in V'} y_i \tag{4.5}$$

$$\sum_{j\in V'} f_{nj} = mp \tag{4.6}$$

$$f_{ij} + f_{ji} = px_{ij} \qquad \forall \{v_i, v_j\} \in \overline{E} \tag{4.7}$$

$$f_{ij} \geq 0, f_{ji} \geq 0 \qquad \forall \{v_i, v_j\} \in \overline{E} \tag{4.8}$$

$$y_i = 1 \qquad \forall v_i \in T \setminus \{v_0\} \tag{4.9}$$

$$x_{ij} = 0, 1 \qquad \forall \{v_i, v_j\} \in \overline{E} \tag{4.10}$$

$$y_i = 0, 1 \qquad \forall v_i \in V' \tag{4.11}$$

$$m \in \mathbb{N}. \tag{4.12}$$

Le fonction objectif (4.1) est de minimiser le coût total de parcours. Les contraintes (4.2) assurent que chaque client de W est couvert, tandis que les contraintes (4.3) as-

surent que chaque nœud de V' est visité au plus une fois. Les contraintes (4.4)-(4.7) définissent les variables de flux. Spécifiquement, les contraintes (4.4) indiquent que le flux d'entrée moins le flux de sortie à chaque nœud $v_i \in V'$ est égal à 2 si v_i est utilisé et à 0 sinon. Le flux de sortie au nœud source v_0 (4.5) est égal à la demande totale de nœuds qui sont utilisés dans la solution, et le flux de sortie à v_n (4.6) correspond à la capacité totale du flux de véhicules. La contrainte (4.7) est dérivée à partir de la définition de variables de flux. Les contraintes (4.10) et (4.12) définissent les variables.

Le figure 4.1 illustre une solution réalisable de mCTP avec deux routes dans le cas où $p = 3$ sous la forme de deux marchandises. Les traits pleins dans la figure représentent les flux f_{ij}, tandis que les lignes pointillées représentent les flux f_{ji}.

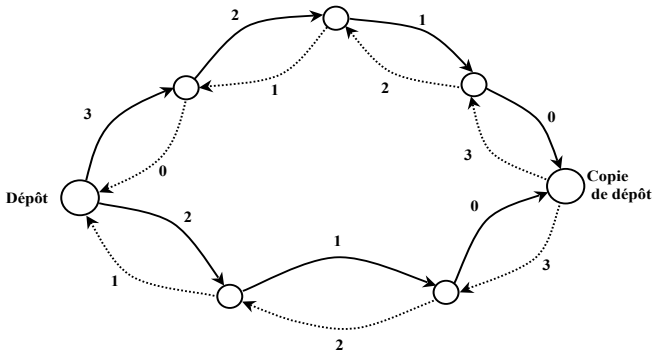

FIGURE 4.1 – Les chemins de flux pour la solution avec deux routes et p=3

Un désavantage de cette formulation est qu'elle ne peut pas exprimer la contrainte sur la longueur de chaque route. Cependant, son avantage est que le nombre de variables et de contraintes augmente de manière polynomiale avec la taille du problème.

4.2 Méthodes de résolution proposées

4.2.1 Algorithme de branchement et coupes

Avant de décrire notre algorithme de branchement et coupes, nous présentons les inégalités valides pour le mCTP. Les inégalités valides pour le CTP peuvent s'appliquer directement à notre problème (voir Gendreau et al. [23]). Dans les inégalités de dominance suivantes (4.14) et (4.15), le nœud v_i est considéré dominer v_j si v_i peut couvrir tous les nœuds de W que v_j peut couvrir.

$$x_{ij} \leqslant y_i \text{ et } x_{ij} \leqslant y_j \ (v_i \text{ ou } v_j \in V \setminus T) \tag{4.13}$$

$$y_i + y_j \leqslant 1 \text{ si } v_i \text{ domine } v_j \text{ ou inversement } (v_i, v_j \in V \setminus T) \tag{4.14}$$

$$y_i + y_j + y_k \leqslant 2 \text{ si deux nœuds parmi } v_i, v_j, v_k \text{ dominent l'autre } (v_i, v_j, v_k \in V \setminus T) \tag{4.15}$$

Toutes les inégalités valides pour le polytope de la couverture d'ensemble (ou "set covering polytope" en anglais) $conv\{x : \sum b_a.y_a \geq 1, y_a \in \{0,1\}\}$, où b_a

sont des coefficients binaires, sont aussi valides pour le mCTP. Balas et Ng [3] ont proposé des facettes avec les coefficients dans {0,1,2} et Sánchez-García et al. [53] ont introduit des facettes plus complexes avec les coefficients dans {0,1,2,3}. Ici, nous rappelons le premier type qui a été utilisé dans Gendreau et al. [23].

Soit S un ensemble non vide de W et définissons pour chaque $v_k \in V$ le coefficient

$$\alpha_k^S = \begin{cases} 0 \text{ si } \lambda_{kl} = 0 \text{ pour tout } w_l \in S, \\ 2 \text{ si } \lambda_{kl} = 1 \text{ pour tout } w_l \in S, \\ 1 \text{ sinon .} \end{cases}$$

L'égalité suivante est alors valide pour le mCTP :

$$\sum_{v_k \in V} \alpha_k^S y_k \geq 2 \tag{4.16}$$

Les inégalités de flux suivantes ont été introduit dans Baldacci et al. [5] :

$$f_{ij} \geq x_{ij}, f_{ji} \geq x_{ji} \text{ si } i, j \neq v_0 \text{ et } i, j \neq v_n \tag{4.17}$$

Nous proposons d'abord de résoudre le mCTP à l'optimalité par un algorithme de branchement et coupes classique. La technique principale est que nous résolvons un programme linéaire contenant les contraintes (4.1), (4.2), (4.3), (4.4), (4.5), (4.6), (4.7), (4.8) et (4.9). Ensuite, nous cherchons des contraintes qui violent des inégalités valides (4.13), (4.14), (4.15), (4.16) et (4.17). Les contraintes détectées sont ajoutées au programme actuel, qui est ensuite ré-optimisé. Ce processus est répété jusqu'à ce que toutes les contraintes soient satisfaites. S'il y a des variables fractionnaires, nous branchons pour générer deux nouveaux sous-problèmes. Si toutes les variables sont entières, nous exploitons un autre sous-problème. L'implémentation de l'algorithme est décrite en détail dans l'article 3 à la fin de ce chapitre.

4.2.2 Une métaheuristique pour le mCTP

Nous introduisons ensuite une métaheuristique qui donne de bonnes solutions pour le mCTP. Elle est utilisée aussi pour fournir des bornes supérieures initiales pour notre algorithme de branchement et coupes.

Dans ce paragraphe, nous ne décrivons que les caractéristiques principales de la métaheuristique (voir l'article 3 à la fin de ce chapitre pour le détail). Notre métaheuristique est un algorithme à deux phases pour le mCTP. Le but de la première phase est de générer aléatoirement nt sous-ensembles de V tels que chaque sous-ensemble puisse couvrir tous les clients. Les nœuds dans chaque sous-ensemble combinés à T créent un ensemble N de nœuds requis qui doivent être visités. Le problème devient maintenant un CVRP bien connu avec les demandes unitaires et est résolu par un algorithme basé sur la méthode de recherche locale évolutive (ELS) de Prins [49] dans la deuxième phase.

Pour générer aléatoirement des sous-ensembles de nœuds couvrant tous les nœuds de W dans la première phase, nous résolvons nt problèmes de PLNE comme suit :

$$\text{Minimiser} \quad \sum_{v_i \in V \setminus T} b_i y_i \tag{4.18}$$

$$\text{sous les contraintes} \quad \sum_{v_i \in V \setminus T} \lambda_{il} y_i \geq 1 \qquad \forall w_l \in W \tag{4.19}$$

$$y_i = 0, 1 \qquad \forall v_i \in V \setminus T \tag{4.20}$$

où b_i est un nombre aléatoire de {1,2}.

Dans la deuxième phase, nous résolvons le problème de CVRP. Nous appliquons la méthode ELS proposée dans Prins [49] à cause de sa simplicité, sa rapidité et son efficacité. Dans la méthode ELS, une solution unique est mutée pour obtenir quelques enfants qui sont ensuite améliorés par la recherche locale. La génération suivante est la meilleure solution entre les parents et ses enfants.

Dans cette phase, une solution est représentée par un tour géant ignorant la capacité des véhicules. Elle est évaluée par la procédure de découpage *Split* de l'heuristique de Beasley [8]. Rappelons que *Split* calcule la meilleure solution possible du CVRP, sous contrainte d'un ordre imposé des nœuds. Pour générer les nouveaux enfants, nous mutons une solution réalisable en échangeant la position d'une ou de plusieurs paires de nœuds.

Outre les recherches locales classiques (voir l'article 3 à la fin de ce chapitre pour le détail), nous utilisons aussi deux nouveautés : un mouvement de saturation (LS1) qui combine deux tournées "non saturées" et un mouvement de nouveau nœud (LS4) qui remplace un nœud dans la solution actuelle par un nouveau nœud.

Recherche locale LS1. Après avoir découpé le tour géant, quelques tournées peuvent être "non saturées" et peuvent être combinées avec d'autres tournées si le nombre de nœuds dans les deux tournées est plus petit ou égal à p. Autrement dit, cette technique nous permet de réduire le nombre de tournées. Pour combiner deux tournées (si possible), nous considérons quatre façons comme illustrées à figure 4.2 et choisissons celle qui nous donne la meilleure économie.

Recherche locale LS4. Cette technique remplace un nœud dans le tour par un nouveau nœud qui n'est pas présent dans la solution actuelle si cela ne viole pas la contrainte de couverture et donne une économie (voir figure 4.3).

4.3 Evaluation numérique

4.3.1 Implémentation et jeux de donnés

Nos algorithmes sont programmés en C/C++ et exécutés sur un CPU Intel Xeon 2,4 GHz. Notons que Jozefowiez [35] a utilisé un CPU Intel Dual Core 2,5 GHz pour son algorithme. Puisqu'un CPU Intel Xeon comporte deux cœurs, nous pouvons estimer que la vitesse des deux machines est identique. Notre algorithme de branchement et coupes est construit autour de CPLEX 11.2 en utilisant la Callable Library.

Nous utilisons la même procédure que Jozefowiez [35] pour générer les instances pour le mCTP. Les instances kroA100, kroB100, kroC100 et kroD100 du TSPLIB sont d'abord utilisées pour créer un ensemble de nb_{total} = V + W = 100 nœuds. Les tests sont réalisés pour $n = \lceil 0.25 nb_{total} \rceil$ et $\lceil 0.5 nb_{total} \rceil, |T| = 1, \lceil 0.20 n \rceil$

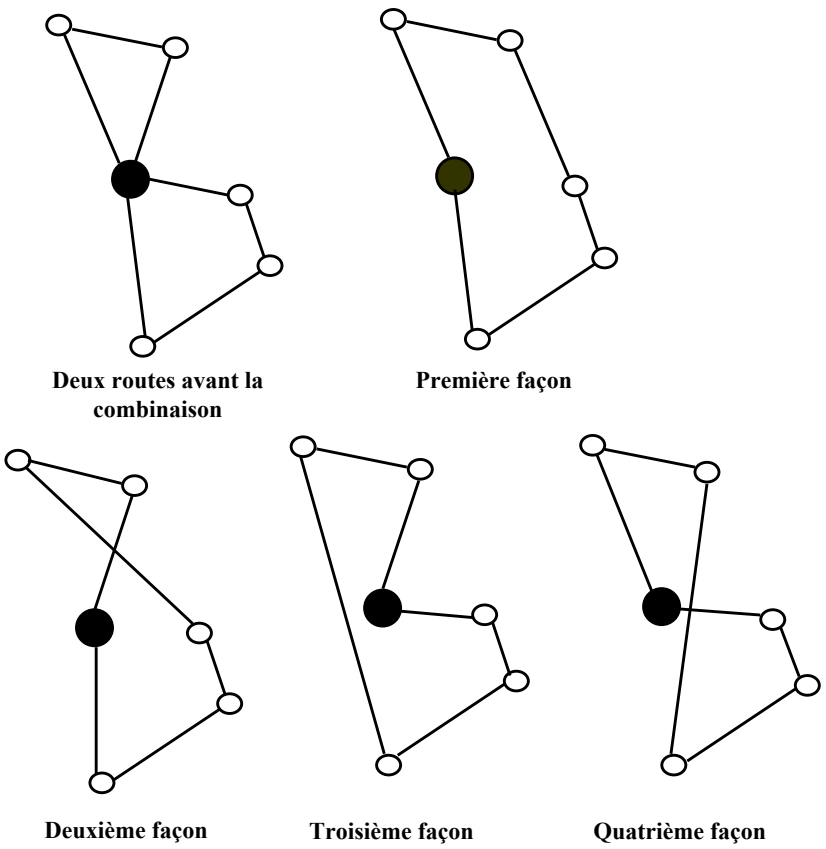

FIGURE 4.2 – Quatre façons de combiner 2 tournées dans LS1

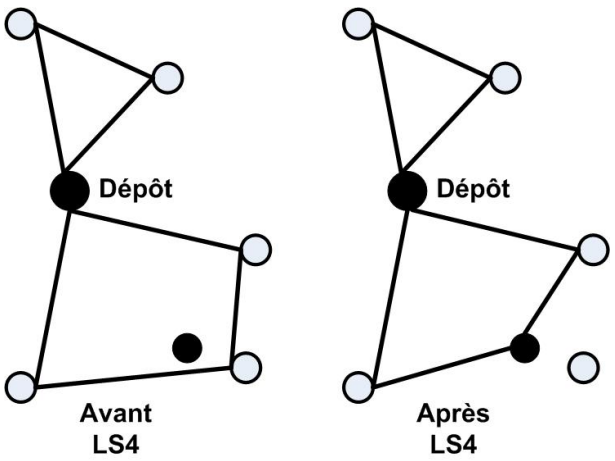

FIGURE 4.3 – Recherche locale LS4

et W contient les points qui restent. Les coûts c_{ij} sont calculés par la distance euclidienne entre deux points. La valeur de c est telle que chaque nœud de $V \setminus T$ couvre au moins un nœud de W et chaque nœud de W est couvert par au moins un nœud de $V \setminus T$ (Voir Gendreau et al. [23], Jozefowiez [35] pour plus d'information). Nous utilisons les instances kroA200 et kroB200 avec nb_{total} = 200 nœuds pour générer

les instances plus grandes pour le mCTP.

Les instances sont étiquetées par X-T-n-W-p, où X est le nom d'instance du TSPLIB. Par exemple, kroA200-1-50-150-4 indique une instance dérivée de kroA200 du TSPLIB avec 1 nœud requis ($|T| = 1$), 50 nœuds qui peut être visités ($|V| = 50$), 150 nœuds qui doivent être couverts ($|W| = 150$), et $p = 4$.

4.3.2 Résultats de l'évaluation numérique

Dans le cadre de cette thèse, nous n'introduisons que les résultats pour 32 instances proposées par Jozefowiez [35] ($nb_{total} = 100$ et $|T| = 1$). Les résultats complets peuvent être trouvés dans l'article 3 à la fin de ce chapitre.

Le tableau 4.1 résume les résultats pour notre évaluation numérique. Les résultats de Jozefowiez [35] sont donnés dans la colonne "Jozefowiez". Pour une comparaison équitable avec la méthode de Jozefowiez [35], dans l'algorithme de branchement et coupes, nous n'utilisons pas les bornes supérieures fournies par la métaheuristique (la colonne "Ha et al. 1" au tableau 4.1) et le temps de calcul pour chaque instance comme dans Jozefowiez [35] est limité à 3600 s. Nous présentons aussi les résultats pour l'algorithme de branchement et coupes dans le cas où les bornes supérieures sont utilisées pour limiter l'arbre de recherche (la colonne "Ha et al. 2" au tableau 4.1).

Dans le tableau 4.1, les résultats en gras impliquent qu'ils sont démontrés optimaux (cas des méthodes exactes) ou qu'ils sont optimaux (cas de la métaheuristique). Les colonnes "Nœud", "Résultat", "Temps" et "m" présentent successivement le nombre de nœuds dans l'arbre de branchement et coupes, la valeur des solutions, le temps de calcul de chaque algorithme et le nombre de véhicules dans la solution.

Comme nous pouvons observer dans ce tableau, notre méthode est clairement meilleure que celle de Jozefowiez [35]. Notre algorithme de branchement et coupes peut résoudre les 32 instances tandis que l'algorithme de Jozefowiez [35] ne peut pas résoudre 10 des instances. Pour les instances qui sont résolues avec succès, notre méthode est aussi plus rapide sur presque toutes les instances.

La version de l'algorithme de branchement et coupes dans le cas où les bornes supérieures initiales fournies par notre métaheuristique sont intégrées est encore meilleure. Par rapport avec la version sans les bornes supérieures, le nombre de nœuds dans l'arbre de branchement et coupes est maintenant plus petit dans 25 instances, le temps de calcul est plus rapide dans 27 sur 32 instances de Jozefowiez [35].

Les résultats confirment la qualité de la métaheuristique. Sur 32 instances de Jozefowiez [35], notre métaheuristique peut trouver 31 solutions optimales. Le gap à l'optimalité de l'instance sans succès est assez petit : 0,11%. De plus, le temps de calcul est rapide : moins de 1 seconde pour toutes les instances.

4.4 Conclusion

Dans ce chapitre, nous avons formulé et résolu un cas particulier de mCTP où la contrainte sur la longueur de chaque tournée est relaxée. Une formulation de programmation linéaire en nombres entiers a été proposée et résolue par un algorithme de branchement et coupes. Une métaheuristique basée sur le principe de ELS a été aussi développée. Les résultats obtenus sur un ensemble d'instances jusqu'à 200

Instance	Ha et al. 1			Jozefowiez		Ha et al. 2			Métaheuristique		
	Nœud	Temps	Résultat	Temps	Résultat	Nœud	Temps	Résultat	m	Temps	Résultat
kroA100-1-25-75-4	10	1.23	**8479**	8	**8479**	7	1.13	**8479**	2	0.16	**8479**
kroA100-1-25-75-5	154	4.20	**8479**	10	**8479**	111	3.27	**8479**	2	0.17	**8479**
kroA100-1-25-75-6	628	10.90	**8479**	9	8724	441	6.87	**8479**	2	0.16	**8479**
kroA100-1-25-75-8	1180	13.32	**7985**	9	**7985**	1827	20.10	**7985**	1	0.16	**7985**
kroA100-1-50-50-4	280	16.88	**10271**	252	**10271**	211	9.91	**10271**	3	0.80	**10271**
kroA100-1-50-50-5	250	15.67	**9220**	1156	**9220**	249	12.36	**9220**	2	0.78	**9220**
kroA100-1-50-50-6	1922	47.41	**9130**	1515	**9130**	1223	24.79	**9130**	2	0.81	**9130**
kroA100-1-50-50-8	13345	228.61	**9130**	3600	11375	12370	203.93	**9130**	1	0.81	**9130**
kroB100-1-25-75-4	30	2.28	**7146**	4	**7146**	20	1.81	**7146**	2	0.22	**7146**
kroB100-1-25-75-5	170	5.14	**6901**	8	**7013**	87	3.23	**6901**	2	0.18	**6901**
kroB100-1-25-75-6	115	4.65	**6450**	10	**6450**	102	4.33	**6450**	1	0.23	**6450**
kroB100-1-25-75-8	655	12.89	**6450**	11	**6450**	593	10.88	**6450**	1	0.20	**6450**
kroB100-1-50-50-4	360	21.35	**10107**	2297	**10107**	306	16.63	**10107**	2	0.62	**10107**
kroB100-1-50-50-5	3655	86.03	**9723**	2038	**9723**	3642	84.08	**9723**	2	0.64	**9723**
kroB100-1-50-50-6	10970	229.12	**9382**	3600	9529	8941	162.24	**9382**	2	0.58	**9382**
kroB100-1-50-50-8	3026	92.07	**8348**	3600	8701	3862	76.06	**8348**	2	0.58	**8348**
kroC100-1-25-75-4	22	2.61	**6161**	3	**6161**	21	2.82	**6161**	1	0.16	**6161**
kroC100-1-25-75-5	149	5.24	**6161**	2	**6161**	177	5.81	**6161**	1	0.16	**6161**
kroC100-1-25-75-6	496	9.23	**6161**	3	**6161**	289	7.73	**6161**	1	0.15	**6161**
kroC100-1-25-75-8	1050	13.87	**6161**	2	**6161**	577	9.42	**6161**	1	0.17	**6161**
kroC100-1-50-50-4	504	24.69	**11372**	174	12156	258	8.12	**11372**	3	0.64	**11372**
kroC100-1-50-50-5	270	12.76	**9900**	1258	**9900**	354	13.28	**9900**	2	0.67	**9900**
kroC100-1-50-50-6	2915	65.35	**9895**	2169	10894	2671	56.91	**9895**	2	0.67	**9895**
kroC100-1-50-50-8	114	11.72	**8699**	3450	8699	109	8.47	**8699**	2	0.65	**8699**
kroD100-1-25-75-4	12	1.33	**7671**	3	**7671**	13	1.04	**7671**	2	0.16	**7671**
kroD100-1-25-75-5	288	5.50	**7465**	15	7759	256	5.38	**7465**	2	0.16	**7465**
kroD100-1-25-75-6	147	4.87	**6651**	13	**6651**	89	3.80	**6651**	1	0.15	**6651**
kroD100-1-25-75-8	1713	24.10	**6651**	10	**6651**	1037	12.85	**6651**	1	0.16	**6651**
kroD100-1-50-50-4	126	14.00	**11606**	239	**11606**	95	9.34	**11606**	3	0.93	**11606**
kroD100-1-50-50-5	882	38.22	**10770**	1562	**10770**	938	29.32	**10770**	2	0.85	**10770**
kroD100-1-50-50-6	7659	175.25	**10525**			12461	281.28	**10525**	2	0.82	10680
kroD100-1-50-50-8	5698	132.04	**9361**	3600	11703	5222	110.62	**9361**	2	0.93	**9361**

Tableau 4.1 – Résultats des algorithmes pour le mCTP

nœuds en tout, et un maximum de 100 nœuds sur le tour ont été présentés et analysés. Ils montrent clairement la performance de nos algorithmes. Notre algorithme de branchement et coupes est meilleur que l'algorithme de Jozefowiez [35] et la solution donnée par la métaheuristique est à moins de 1,45% de l'optimalité pour l'ensemble des instances testées (voir l'article 3 à la fin de ce chapitre).

L'étape suivante de notre recherche est d'améliorer l'algorithme de branchement et coupes en ajoutant plus de coupes efficientes afin de mieux évaluer notre métaheuristique sur les instances plus grandes. Une autre direction de recherche intéressante est de chercher un mécanisme de rétroaction de la deuxième phase vers la première phase, par exemple chercher une manière de fixer plus intelligemment des coefficients b_i dans la programmation de PLNE de la première phase basée sur l'information de la deuxième phase.

Article 3 : An Exact Algorithm and a Metaheuristic for the multi-vehicle Covering Tour Problem with a Constraint on the Number of Vertices

Minh Hoàng Hà, Nathalie Bostel, André Langevin, Louis-Martin Rousseau, European Journal of Operational Research, Volume 226, Issue 2, Pages 211-220.

An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices

Minh Hoang Ha[a,c], Nathalie Bostel[b], André Langevin[c,*], Louis-Martin Rousseau[c]

[a] *L'UNAM Université, École des Mines de Nantes, IRCCyN UMR CNRS 6597, 4 rue Alfred Kastler, 44307 Nantes Cedex 3, France*
[b] *L'UNAM Université, Université de Nantes, IRCCyN UMR CNRS 6597, 58 rue Michel Ange, B.P. 420, 44606 Saint-Nazaire Cedex, France*
[c] *Department of Mathematics and Industrial Engineering and CIRRELT, École Polytechnique de Montréal, C.P. 6079, Succursale Centre-ville, Montréal, QC, Canada H3C 3A7*

Abstract

The multi-vehicle covering tour problem (m-CTP) involves finding a minimum-length set of vehicle routes passing through a subset of vertices, subject to constraints on the length of each route and the number of vertices that it contains, such that each vertex not included in any route lies within a given distance of a route. This paper tackles a particular case of m-CTP where only the restriction on the number of vertices is considered, i.e., the constraint on the length is relaxed. The problem is solved by a branch-and-cut algorithm and a metaheuristic. To develop the branch-and-cut algorithm, we use a new integer programming formulation based on a two-commodity flow model. The metaheuristic is based on the evolutionary local search (ELS) method proposed in [23]. Computational results are reported for a set of test problems derived from the TSPLIB.

Keywords: Covering tour, Two-commodity flow model, Branch-and-cut, Metaheuristic

*Corresponding author: Tel.: +1-514-340-4711x4511 fax: +1-514-340-4463
Email address: andre.langevin@polymtl.ca (André Langevin)

Preprint submitted to European Journal of Operational Research

1. Introduction

The *multi-vehicle covering tour problem* (m-CTP) is a generalization of the *vehicle routing problem* (VRP), which is an extension of the *covering tour problem* (CTP). The m-CTP is defined as follows.

Let $G = (V \cup W, E_1 \cup E_2)$ be an undirected graph, where $V \cup W$ is the vertex set and $E_1 \cup E_2$ is the edge set. $V = \{v_0, ..., v_{n-1}\}$ is the set of n vertices that can be visited, and $W = \{w_1, w_2, ..., w_l\}$ is the set of vertices that must be covered. Let $T = \{v_0, ..., v_{t-1}\}$, a subset of V, be the set of vertices that must be visited. Vertex v_0 is the depot; m identical vehicles are located there. This paper considers the case where m is a decision variable. A length c_{ij} is associated with each edge of $E_1 = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ and a distance d_{ij} is associated with each edge of $E_2 = \{(v_i, v_j) : v_i \in V \setminus T, v_j \in W\}$. The m-CTP consists in finding m vehicle routes such that the total cost is minimized and

- Each route begins and ends at the depot;
- Each vertex of T is visited exactly once while each vertex of $V \setminus T$ is visited at most once;
- Each vertex j of W is covered by the routes, i.e., lies within a distance r of at least one vertex of $V \setminus T$ that is visited, where r is the covering radius;
- The number of vertices on each route (excluding the depot) is less than a given value p;
- The length of each route does not exceed a fixed limit q.

The m-CTP is clearly NP-hard since it reduces to a VRP with unit demands when$T = V$ and $W = \emptyset$ or to a CTP when the capacity constraints are relaxed.

The m-CTP can model problems concerned with the design of bilevel transportation networks, such as the construction of routes for mobile health-care teams (see, e.g., [15, 28]) and mobile library teams, and the location of post boxes [18], banking agencies, and milk collection points [27]. Recently, the model of the CTP has also been used to solve the disaster relief problem in [8]. In this application, after a disaster the health care organizations have to supply the affected populations with food, water and medicine. The relief

vehicles (e.g. mobile hospitals) stop at several locations and the populations (the set W in the mathematical description of the problem) must visit one of the vehicle stops. The health care organizations have to choose the appropriate stops among $|V|$ potential locations so that all populations can reach one of these stops within acceptable time. T can be considered as the set of stops covering the populations that cannot be covered by other stops.

Many researchers have studied classical vehicle routing problems where all the customers have to be served, but the number of papers on the CTP is much more limited. It seems that the first work on this problem can be credited to [7] in 1981. Since then the CTP has received little attention from the research community. The one-vehicle version (1-CTP) was solved exactly by a branch-and-cut algorithm in [10]. A heuristic was also proposed in this paper. The authors of [3] used a two-commodity flow formulation and developed a scatter search algorithm. For the multi-vehicle version, [14] introduced a three-index vehicle flow formulation and three heuristics inspired by classical algorithms: Clarke and Wright [6], the sweep algorithm [11], and the route-first/cluster-second method [5]. The three heuristics are compared to each other, and the optimality gap is therefore unknown. Recently, [16] has proposed the first exact algorithm. It is based on a column generation approach in which the master problem is a simple set covering problem, and the subproblem is formulated similarly to the 1-CTP model of [10]. The algorithm was tested on instances derived from TSPLIB, and results for $|V|+|W|=100$ and $|T|=1$ are reported.

The close-enough arc routing problem (CEARP) can be seen as an arc-routing counterpart of CTP. It is similar to the CTP except that a closed tour must be determined so that every vertex of W lies within a distance r of an *arc* of the tour. This problem was considered in [12, 13].

In this paper, we address a particular case of m-CTP where the length constraint is relaxed, i.e., $q=+\infty$. We refer to this as m-CTP-p (p is the upper bound on the number of vertices per route). This version can model the applications where the distance constraint is not important and can be relaxed. An example is in the vaccination campaigns where the health care team has to vaccinate the populations at several locations. When the travelling time among the locations is quite small compared with the service duration at a location, we can consider that each vehicle can serve only a limited number of locations and the distance constraint in this case can be neglected. Our contributions are that we present a new formulation for m-CTP-p and propose an exact method for this problem, as well as a metaheuristic. Computational

experiments show that our exact approach outperforms the column generation method of [16], and our metaheuristic gives high-quality solutions for the tested instances.

The remainder of the paper is organized as follows. Section 2 describes our formulation and several valid inequalities. The branch-and-cut algorithm and metaheuristic are presented in Sections 3 and 4 respectively. Section 5 discusses the computational results, and Section 6 summarizes our conclusions.

2. New formulation for *m*-CTP-*p*

In this section, we describe a new integer programming formulation for m-CTP-p. The idea underlying this formulation was first introduced by [9] for the traveling salesman problem (TSP). Langevin et al. [19] extended this approach to solve the TSP with time windows. Baldacci et al. [4] used this method to derive a new formulation and a branch-and-cut for the VRP, and Baldacci et al. [3] adapted it to formulate the 1-CTP without the capacity constraints.

Our formulation is an extension of that proposed by Baldacci et al. [3] for the 1-CTP. To adapt this idea for m-CTP-p, we consider that each vertex of $V \setminus \{v_0\}$ has a unit demand and each vehicle has a capacity of p. We also note that the difference between m-CTP-p and VRP is that we do not need to visit all vertices of V with the exception of the vertices of T.

The original graph G is first extended to $\overline{G} = (\overline{V} \cup W, \bar{E}_1 \cup E_2)$ by adding a new vertex v_n, which is a copy of the depot v_0. We have $\overline{V} = V \cup \{v_n\}$, $V' = \overline{V} \setminus \{v_0, v_n\}$, $\overline{E} = E_1 \cup \{(v_i, v_n), v_i \in V'\}$, and $c_{in} = c_{0i} \; \forall \; v_i \in V'$.

This formulation requires two flow variables, f_{ij} and f_{ji}, to represent an edge of a feasible m-CTP-p solution along which the vehicle initially carries a load of p units. When a vehicle travels from v_i to v_j, flow f_{ij} represents the number of vertices that can still be visited and flow f_{ji} represents the number of vertices already visited (i.e., $f_{ji} = p$ - f_{ij}).

Let x_{ij} be a 0-1 variable equal to 1 if edge $\{v_i, v_j\}$ is used in the solution and 0 otherwise. Let y_i be a binary variable that indicates the presence of vertex v_i in the solution. We set the binary coefficients λ_{il} equal to 1 if and only if $w_l \in W$ can be covered by $v_i \in V \setminus T$. Then m-CTP-p can be stated as:

$$\text{Minimize} \quad \sum_{\{v_i,v_j\}\in\overline{E}} c_{ij}x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{v_i\in V\setminus T} \lambda_{il}y_i \geq 1 \qquad \forall w_l \in W \tag{2}$$

$$\sum_{v_i\in\overline{V},i<k} x_{ik} + \sum_{v_j\in\overline{V},j>k} x_{kj} = 2y_k \qquad \forall v_k \in V' \tag{3}$$

$$\sum_{v_j\in\overline{V}} (f_{ji} - f_{ij}) = 2y_i \qquad \forall v_i \in V' \tag{4}$$

$$\sum_{v_j\in V'} f_{0j} = \sum_{v_i\in V'} y_i \tag{5}$$

$$\sum_{j\in V'} f_{nj} = mp \tag{6}$$

$$f_{ij} + f_{ji} = px_{ij} \qquad \forall\{v_i,v_j\} \in \overline{E} \tag{7}$$

$$f_{ij} \geq 0, f_{ji} \geq 0 \qquad \forall\{v_i,v_j\} \in \overline{E} \tag{8}$$

$$y_i = 1 \qquad \forall v_i \in T \setminus \{v_0\} \tag{9}$$

$$x_{ij} \in \{0,1\} \qquad \forall\{v_i,v_j\} \in \overline{E} \tag{10}$$

$$y_i \in \{0,1\} \qquad \forall v_i \in V' \tag{11}$$

$$m \in \mathbb{N}. \tag{12}$$

The objective (1) is to minimize the total travel cost. Constraints (2) ensure that every customer of W is covered, while constraints (3) ensure that each vertex of V' is visited at most once. Constraints (4) to (7) define the flow variables. Specifically, constraints (4) state that the inflow minus the outflow at each vertex $v_i \in V'$ is equal to 2 if v_i is used and to 0 otherwise. The outflow at the source vertex v_0 (5) is equal to the total demand of the vertices that are used in the solution, and the outflow at the sink v_n (6) corresponds to the total capacity of the vehicle fleet. Constraint (7) is derived from the definition of the flow variables. Constraints (10) and (12) define the variables.

Figure 1 shows a feasible solution of m-CTP-p with two routes in the case where $p = 3$ under the two-commodity form. The solid lines in the figure represent the flows f_{ij} while the dotted lines represent the flows f_{ji}.

A disadvantage of this formulation is that it can not express the constraint

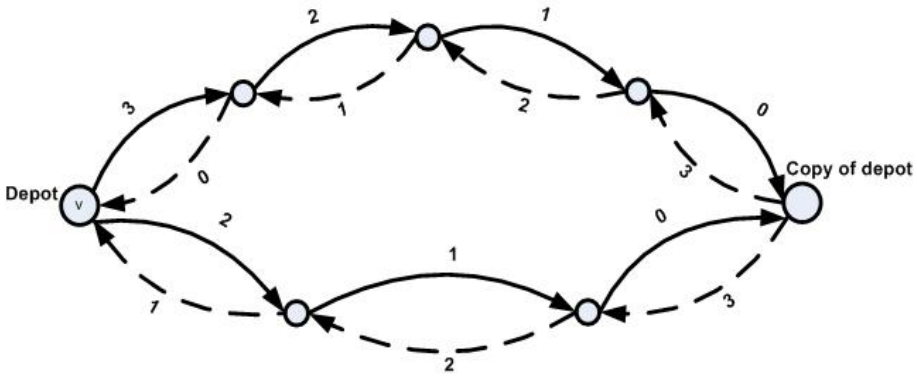

Figure 1: Flow paths for solution with two routes and p=3

on the length of each route. However, its advantages are that the number of variables and constraints increases polynomially with the size of the problem, and its LP relaxation satisfies a weak form of the subtour elimination constraints (see [4]).

The linear relaxation of m-CTP-p can be strengthened by the addition of valid inequalities. The valid inequalities for 1-CTP apply directly to our problem (see [10]). In the following dominance inequalities (14), a vertex v_i is said to dominate v_j if v_i can cover all the vertices of W that v_j can cover. In the dominance inequalities (15), this dominance relation is extended to three vertices where a subset of two vertices dominates the third vertex. We have

$$x_{ij} \leqslant y_i \text{ and } x_{ij} \leqslant y_j \ (v_i \text{ or } v_j \in V \setminus T) \tag{13}$$

$$y_i + y_j \leqslant 1 \text{ if } v_i \text{ dominates } v_j \text{ or conversely } (v_i, v_j \in V \setminus T) \tag{14}$$

$$y_i + y_j + y_k \leqslant 2 \text{ if two of } v_i, v_j, v_k \text{ dominate the other vertex } (v_i, v_j, v_k \in V \setminus T). \tag{15}$$

All the valid inequalities of the set covering polytope $conv\{y : \sum b_a.y_a \geq 1,\ y_a \in \{0,1\}\}$ where b_a is the binary coefficient, are valid for m-CTP-p. Balas and Ng [2] introduced the facets with coefficients in {0,1,2} and Sánchez-García et al. [26] introduced the more complex facets with coefficients in {0,1,2,3}. Here, we recall the first one that was used in [10]: let S be a nonempty subset of W, and define for each $v_k \in V$ the coefficient

$$\alpha_k^S = \begin{cases} 0 \text{ if } \lambda_{kl} = 0 \text{ for all } w_l \in S, \\ 2 \text{ if } \lambda_{kl} = 1 \text{ for all } w_l \in S, \\ 1 \text{ otherwise.} \end{cases}$$

Then the following constraint is valid for m-CTP-p:

$$\sum_{v_k \in V} \alpha_k^S y_k \geq 2. \tag{16}$$

The following flow inequalities were introduced in [3]:

$$f_{ij} \geq x_{ij}, f_{ji} \geq x_{ji} \text{ if } i, j \neq v_0 \text{ and } i, j \neq v_n. \tag{17}$$

Several other valid inequalities for the VRP expressed for the set T of required vertices can be applied directly to our problem. Here we restrict ourselves to the capacity constraints, originally proposed by [20]:

$$\sum_{(v_i, v_j \in S)} x_{ij} \leq |S| - \left\lceil \frac{|S|}{p} \right\rceil (S \subseteq T, |S| \geq 2). \tag{18}$$

Let z be the minimum number of vertices required to cover all vertices of W. The following constraint follows immediately:

$$m \geq \left\lceil \frac{z}{p} \right\rceil. \tag{19}$$

The value of z can be calculated by solving a set covering problem as follows:

$$\text{Minimize} \qquad z = |T| + \sum_{(v_i \in V \setminus T)} y_i \tag{20}$$

$$\text{subject to} \qquad \sum_{v_i \in V \setminus T} \lambda_{il} y_i \geq 1 \qquad \forall w_l \in W \tag{21}$$

$$y_i = 0, 1 \qquad \forall v_i \in V \setminus T. \tag{22}$$

3. Branch-and-cut algorithm

We solve m-CTP-p exactly using a standard branch-and-cut algorithm. We solve a linear program containing the constraints (1), (2), (3), (4), (5), (6), (7), (8), (9), and (19). We then search for violated constraints of type (13), (14), (15), (16), (17), and (18), and the detected constraints are added to the current LP, which is then reoptimized. This process is repeated until all the constraints are satisfied. If there are fractional variables, we branch. If all the variables are integer, we explore another node.

The separation of the constraints of type (13), (14), (15), and (17) is straightforward. For constraints (16), as in [10], to reduce the computational effort we verify only the sets S that include three elements.

To generate the capacity constraints (18), we use the *greedy randomized algorithm*, proposed by [1] and reused in [4]. This is an iterative procedure that is applied to subsets $T' \subset T$ created a priori. At each iteration, the following procedure is repeated for each $S \in T'$. Let $v_{i^*} \in T \setminus S$ be a vertex such that

$$\sum_{j \in S} (x_{i^* j} + x_{j i^*}) = \max_{i \in T \setminus S} \left[\sum_{j \in S} (x_{ij} + x_{ji}) \right].$$

If the current solution x violates the capacity constraints (18) corresponding to the subset $S' = S \cup i^*$, then we add this inequality to the model, update S to S', and repeat the process until S contains all vertices of T. In our implementation, the initial set T' can be a single vertex of T because the number of vertices is fairly small in m-CTP-p instances.

Our branch-and-cut algorithm is built around CPLEX 11.2 with the Callable Library. We tested the algorithm with the activation of each CPLEX cut one by one, and observed that only two of them (Gomory cut and implied-bound cut) were useful. Thus, all CPLEX cuts except the Gomory and implied-bound cuts are turned off. Gomory cuts are generated by applying integer rounding on a pivot row in the optimal LP tableau for a (basic) integer variable with a fractional solution value. These cuts are applied to solve the VRP (see [21] for example). Implied-bound cuts are generated in some models where binary variables imply bounds on continuous variables. Unfortunately, we do not know why the implied-bound cuts are useful. All the other CPLEX parameters are set to their default values.

We tested several branching techniques, such as branching on the variables y before x as in [10] and branching on the variables x before y, but these do not outperform the CPLEX branching. Hence, we let CPLEX make the branching decisions.

4. Metaheuristic

In this section, we introduce a metaheuristic for m-CTP-p that is a two-phase hybrid algorithm. The aim of the first phase is to randomly generate nt subsets of V such that each subset can cover all the customers. The vertices of each subset combined with T create a set N of the vertices that must be visited. The problem now becomes a VRP with unit demands, and it is solved in the second phase by an algorithm based on the ELS method of [23].

4.1. First phase

To randomly generate subsets of vertices covering all vertices of W, we solve the following θ_1 mixed integer programming problems:

$$\text{Minimize} \qquad \sum_{v_i \in V \setminus T} b_i y_i \tag{23}$$

$$\text{subject to} \qquad \sum_{v_i \in V \setminus T} \lambda_{il} y_i \geq 1 \qquad \forall w_l \in W \tag{24}$$

$$y_i = 0, 1 \qquad \forall v_i \in V \setminus T \tag{25}$$

where b_i is a random number in $\{1,2\}$. The solution of this model is rapid even for the large instances in our tests.

4.2. Second phase

The goal of the second phase is to solve the VRP problems. We apply the ELS method proposed in [23] because of its simplicity, speed, and good performance. In the ELS method, a single solution is mutated to obtain several children that are then improved by local search. The next generation is the best solution among the parent and its children. We now introduce the procedures for the construction of the second phase.

4.2.1. Split, concat, and mutate procedures

The split procedure is the backbone of our metaheuristic. Its goal is to split a giant tour into VRP routes. This procedure was originally introduced in [5] and then integrated into memetic algorithms to successfully solve various vehicle routing problems (see [17, 24, 22] for example). The reader is referred to [25] for an efficient implementation of this procedure. The concat procedure does the opposite: it concatenates VRP routes into a giant tour.

The output of the mutate(L) procedure is a randomly perturbed copy L' of the input giant tour L. This procedure randomly swaps the position of two vertices in the original tour.

4.2.2. Local search procedures

As in [23], we can use classical moves such as 2-opt moves, Or-opt moves, or string-exchange moves. Here, we use two simple classical local searches: relocation of a node (LS2) and two-point moves that swap the position of two nodes (LS3). We also introduce two new approaches: saturation moves (LS1) that combine two unsaturated routes, and new-node moves (LS4) that try to replace a node in the solution by a new node.

Local search LS1. After we split the giant tour, some routes may not be saturated, and they can be combined with other routes if the total number of nodes in the two routes is less than p. In other words, this technique helps us to reduce the number of routes. To combine two routes (if possible), we consider the four methods illustrated in Fig. 2 and choose the best.

Local search LS4. This technique replaces a node in the tour by a new node that is not present in the current solution. The swap is done if it does not violate the covering constraint and improves the solution (see Fig. 3). LS4 is called after each ELS phase. If it can improve the solution, the new solution (i.e., after LS4) becomes the initial solution for a new ELS phase; otherwise, we return to the first phase.

After many tests, we decided to use *first-improvement local search* for LS1 and LS4 and *best-improvement local search* for LS2 and LS3. In first-improvement local search, if an improving move is detected, it is immediately executed and the remaining moves are bypassed. The process is repeated until we can not find a better solution. Best-improvement local search evaluates all the possible moves and executes the best one.

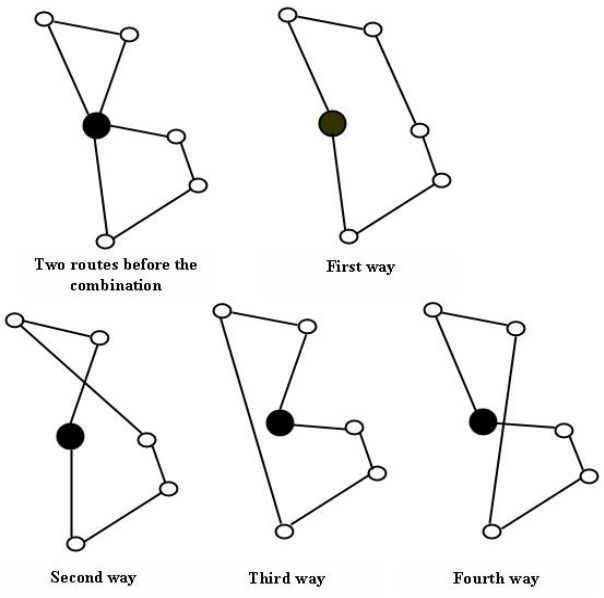

Figure 2: Four ways to combine two routes in LS1

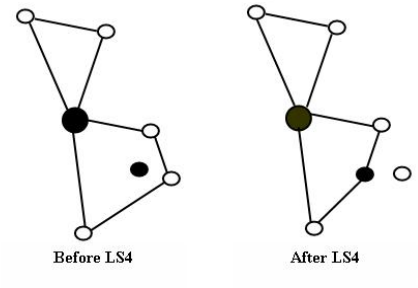

Figure 3: Local search LS4

4.2.3. Initial solution

We use the route-first/cluster-second approach to generate the initial solution. To create giant tours, we use two procedures: insertion of the nearest neighbor (Insert1) and the greatest-saving insertion (Insert2). We split these two tours to obtain two solutions. The initial solution is the best found by local search LS1.

4.2.4. Tabu list

In our algorithm, the tabu list stores attributes of the giant tours instead of final solutions. We use the *insertion of nearest neighbor* procedure to build the tabu list by calculating the length of the giant tour created. The set of nodes that must be visited is now represented by the length of a tour constructed by procedure Insert1. The tabu list is initialized only once and has a length of θ_1, i.e., we will store all θ_1 solutions generated at the first phase. The computational results show that this list helps us to avoid re-exploiting 27.43% of the giant tours on average.

4.3. Resulting algorithm

Algorithm 1 gives the pseudocode for the resulting metaheuristic. Note that because the split procedure can handle the length constraints (see [25] for details), our algorithm can solve the general m-CTP problem.

5. Computational experiments

In this section, we describe the m-CTP-p instances and the computational evaluation of the proposed algorithm. Our algorithm is coded in C/C++ and is run on the same CPU used in [16], i.e., a 2.4-GHz CPU with 4 GB of RAM. The running time of the branch-and-cut algorithm is limited to 2 h for each instance.

The parameters θ_1, θ_2, and θ_3 in the metaheuristic are chosen so that they depend only on the problem data, i.e., they are generated according to an automatic mechanism. We tested many combinations and found that the following combination gives the best performance for our algorithm: $\{\theta_1, \theta_2, \theta_3\}$=$\{5(|V| - |T|), |N|, |N|\}$ where N is the set of required vertices generated by the first phase.

In the tables of results, the blank entries indicate that the algorithm did not find a solution. The column headings are as follows:

Data: name of instance;

Node: number of nodes in search tree of branch-and-cut algorithm;

Time: running time in seconds;

m: number of vehicles in solution;

Nv: number of vertices of V visited by the route in the optimal solution;

Imp: number of implied-bound cuts;

Go: number of Gomory cuts;

Do1: number of constraints of type (13);

Algorithm 1 Pseudocode for metaheuristic

```
1: f(S_final) ⇐ +∞;
2: Tabulist ⇐ ∅;
3: for t = 1 → θ1 do
4:     N ⇐ initialize random MIP generator;
5:     if N ∈ Tabulist then
6:         go to line 4;
7:     else
8:         add N to Tabulist;
9:     end if
10:    L1 ⇐ Insert1(N);
11:    S1 ⇐ Split(L1);
12:    S1 ⇐ LS1(S1);
13:    S* ⇐ S1;
14:    L2 ⇐ Insert2(N);
15:    S2 ⇐ Split(L2);
16:    S2 ⇐ LS1(S1);
17:    if f(S2) < f(S1) then
18:        S* ⇐ S2;
19:    end if
20:    for i = 1 → θ2 do
21:        f̄ ⇐ +∞;
22:        for j = 1 → θ3 do
23:            L ⇐ Concat(S*);
24:            L ⇐ Mutate(L);
25:            S ⇐ Split(L);
26:            S ⇐ LS1(S);
27:            S ⇐ LS2(S);
28:            S ⇐ LS3(S);
29:            if f(S) < f̄ then
30:                f̄ ⇐ f(S);
31:                S̄ ⇐ S;
32:            end if
33:        end for
34:        if f̄ < f(S*) then
35:            S* ⇐ S̄;
36:        end if
37:    end for
38:    S ⇐ S*;
39:    S* ⇐ LS4(S*);
40:    if f(S*) < f(S) then
41:        go to line 20;
42:    end if
43:    if f(S*) < f(S_final) then
44:        S_final ⇐ S*;
45:        f(S_final) ⇐ f(S*);
46:    end if
47: end for
```

Do2: number of constraints of type (14) and (15);
Cov: number of constraints of type (16);
Flow: number of constraints of type (17);
Cap: number of constraints of type (18);
$LB0$: value of lower bounds at the root of the search tree (before adding the cuts);
$LB1$: value of lower bounds at the root of the search tree (after adding the cuts);
LB: the best lower bound in the branch-and-cut tree;
LS1: number of times LS1 is called;
LS2: number of times LS2 is called;
LS3: number of times LS3 is called;
LS4: number of times LS4 is called.

5.1. Data instances

We use the same procedure used in [16] to generate the instances for m-CTP-p. The instances kroA100, kroB100, kroC100, and kroD100 of TSPLIB are first used to create a set of $nb_{total} = V + W = 100$ vertices. Tests are run for $n = \lceil 0.25nb_{total} \rceil$ and $\lceil 0.5nb_{total} \rceil$ and $|T| = 1$ and $\lceil 0.20n \rceil$, and W is defined by taking the remaining points. The c_{ij} are computed as the Euclidean distances between the points. The value of c is determined so that each vertex of $V \setminus T$ covers at least one vertex of W, and each vertex of W is covered by at least two vertices of $V \setminus T$ (see [10, 16] for further information). We also use instances kroA200 and kroB200 with $nb_{total} = 200$ vertices to generate larger instances for m-CTP-p.

The instances are labeled X-T-n-W-p, where X is the name of the TSPLIB instance. For example, A2-1-50-150-4 indicates an instance derived from kroA200 of TSPLIB with 1 required vertex ($|T| = 1$), 50 vertices that can be visited ($|V| = 50$), 150 vertices that must be covered ($|W| = 150$), and $p = 4$.

5.2. Comparison with method of [16]

For a fair comparison with the exact algorithm of [16], we do not use the upper bound provided by the metaheuristic, and the running time for each instance is, as in [16], limited to 3600 s.

Table 1 gives the results of this experiment. The results in bold are proved to be optimal. The results of [16] which in some cases are not in bold (for example, the instance A1-25-75-6) mean that the method of [16] gives

a solution not proven optimal. Jozefowiez gave some non-optimal solutions although the running time did not reach to limit because he developed an algorithm based on the column generation, not a branch-and-price one and the maximum number of columns searched at each iteration was limited to 25.

As can be seen, our method clearly outperforms that of [16]. Our branch-and-cut algorithm can solve all 32 instances, whereas the algorithm of [16] is unable to solve 10 instances. Our method is also faster on almost all of the successfully solved instances.

	Our method				Jozefowiez		
Data	m	Node	Time	Result	m	Time	Result
A1-25-75-4	2	10	1.23	**8479**	2	8	**8479**
A1-25-75-5	2	154	4.20	**8479**	2	10	**8479**
A1-25-75-6	2	628	10.90	**8479**	2	9	8724
A1-25-75-8	1	1180	13.32	**7985**	1	9	**7985**
A1-50-50-4	3	280	16.88	**10271**	3	252	**10271**
A1-50-50-5	2	250	15.67	**9220**	2	1156	**9220**
A1-50-50-6	2	1922	47.41	**9130**	2	1515	**9130**
A1-50-50-8	2	13345	228.61	**9130**	2	3600	11375
B1-25-75-4	2	30	2.28	**7146**	2	4	**7146**
B1-25-75-5	2	170	5.14	**6901**	2	8	**7013**
B1-25-75-6	1	115	4.65	**6450**	1	10	**6450**
B1-25-75-8	1	655	12.89	**6450**	1	11	**6450**
B1-50-50-4	2	360	21.35	**10107**	2	2297	**10107**
B1-50-50-5	2	3655	86.03	**9723**	2	2038	**9723**
B1-50-50-6	2	10970	229.12	**9382**	2	3600	9529
B1-50-50-8	2	3026	92.07	**8348**	1	3600	8701
C1-25-75-4	1	22	2.61	**6161**	1	3	**6161**
C1-25-75-5	1	149	5.24	**6161**	1	2	**6161**
C1-25-75-6	1	496	9.23	**6161**	1	3	**6161**
C1-25-75-8	1	1050	13.87	**6161**	1	2	**6161**
C1-50-50-4	3	504	24.69	**11372**	3	174	12156
C1-50-50-5	2	270	12.76	**9900**	2	1258	**9900**
C1-50-50-6	2	2915	65.35	**9895**	2	2169	10894
C1-50-50-8	2	114	11.72	**8699**	2	3450	8699
D1-25-75-4	2	12	1.33	**7671**	2	3	**7671**
D1-25-75-5	2	288	5.50	**7465**	2	15	7759
D1-25-75-6	1	147	4.87	**6651**	1	13	**6651**
D1-25-75-8	1	1713	24.10	**6651**	1	10	**6651**
D1-50-50-4	3	126	14.00	**11606**	3	239	**11606**
D1-50-50-5	2	882	38.22	**10770**	2	1562	**10770**
D1-50-50-6	2	7659	175.25	**10525**			
D1-50-50-8	2	5698	132.04	**9361**	3	3600	11703

Table 1: Comparison with method of [16]

5.3. Results for branch-and-cut algorithm

This subsection presents the results of the branch-and-cut algorithm in the case where the initial upper bounds provided by our metaheuristic are integrated as the bounds on the objective function. Let UB be the value of the final solution found by branch-and-cut algorithm or the value of the solution of the metaheuristic (if the branch-and-cut algorithm fails to find a solution), Gap_{BnC} in Tables 2 and 3 is computed as:

$$Gap_{BnC} = \frac{100.(UB - LB)}{UB} \tag{26}$$

Table 2 shows that our exact method can solve all but one of the instances with 100 vertices. Compared with the version without initial upper bounds, the number of nodes in the search tree is now lower in 25 of the 32 instances of [16]. For the 32 instances with 200 vertices (see Table 3), our algorithm can solve 20 instances successfully; most of them have $n = 50$. The problem difficulty increases with n and T but is fairly insensitive to $|W|$. This is similar to problem 1-CTP in [10]. Moreover, the greater the value of p, the harder the problem. Instances with $p = 4$ or 5 are usually solved more readily than instances with higher p values.

Tables 4 and 5 present the number of constraints generated in the branch-and-cut algorithm and the lower bounds at the root node of the search tree. In these tables, Gap_{LB} shows the deviation between the lower bounds LB0 and LB1 and is computed as:

$$Gap_{LB} = \frac{100.(LB1 - LB0)}{LB0} \tag{27}$$

Tables 4 and 5 clearly show the performance of valid inequalities in improving the linear relaxation of m-CTP-p, specially on the instances with $|T| = 1$. Among the cuts added, the flow constraints (17) are the most frequent; capacity constraints (18) are generated only when the value of $|T|$ is important.

5.4. Results for metaheuristic

Tables 6 and 7 report our results for the metaheuristic. The numbers in bold indicate the optimal solutions found by the branch-and-cut algorithm. The numbers marked with an asterix correspond to solutions that can not be improved by the branch-and-cut algorithm. Let UB be the value of the solution of the metaheuristic, Gap_{UB} in these tables reports the percentage deviation of the metaheuristic and is computed as:

$$Gap_{UB} = \frac{100.(UB - LB)}{UB} \tag{28}$$

where LB is the best lower bound in the branch-and-cut tree.

Our results confirm the quality of the metaheuristic. For the 82 instances for which the branch-and-cut algorithm can find an optimal solution, our metaheuristic can find an optimal solution in 79 cases. The optimality gaps for the three unsuccessful instances are small: 0.11%, 0.16%, and 1.45%. For the instances whose optimal solution is unknown, the branch-and-cut algorithm can not improve on the metaheuristic solution and we believe that the large gap G_{UB} in some cases are due to the poor quality of the lower bounds LB. Moreover, the running time is acceptable: it has a maximum of 126.00 s on the largest instance, kroB200-20-100-100-5.

6. Conclusion

In this paper, we have formulated and solved a particular case of the m-CTP where the length constraint is relaxed. We have presented an integer linear programming formulation that is solved using a branch-and-cut algorithm and developed a metaheuristic based on the ELS principle. We have reported computational results for a set of instances with up to 200 vertices where the tour contains up to 100 vertices. Our results clearly show the performance of our approach. Our branch-and-cut algorithm outperforms the algorithm of [16], and the solution provided by the metaheuristic is within 1.45% of optimality for the considered test instances.

The next step of our research will be to improve the branch-and-cut algorithm by adding more efficient cuts, so that we can evaluate our metaheuristic on larger instances. One line of investigation will be to study how the implied-bound cuts of CPLEX strengthen the linear relaxation of the model.

References

[1] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem. Rapport de recherche, ARTEMIS-IMAG, Grenoble, France, 1995.

[2] E. Balas and Shu Ming Ng. On the set covering polytope: I. All the facets with coefficients in {0, 1, 2}. *Mathematical Programming*, 43(1-3):57–69, 1986.

[3] R. Baldacci, M. A. Boschetti, V. Maniezzo, and M. Zamboni. Scatter search methods for covering tour problem. In Ramesh Sharda, Stefan Vob, César Rego, and Bahram Alidaee, editors, *Metaheuristic Optimization Via Memory and Evolution*, pages 55–91. Springer Verlag, 2005.

[4] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52:723–738, 2004.

[5] J. E. Beasley. Route first-cluster second methods for vehicle routing. *Omega*, 11:403–408, 1983.

[6] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[7] J. Current. *Multiobjective design of transportation networks*. PhD dissertation, Johns Hopkins University, 1981.

[8] K-F. Doerner and R-F. Hartl. Health care logistics, emergency preparedness, and disater relief: New challenges for routing problems with a focus on the austrian situation. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 527–550. Springer, 2008.

[9] G. A. Finke, A. Claus, and E. Gunn. A two-commodity network flow approach to the traveling salesman problem. *Congressus Numerantium*, 41:167–178, 1984.

[10] M. Gendreau, G. Laporte, and F. Semet. The covering tour problem. *Operations Research*, 45:568–576, 1997.

[11] B. Gillett and L. Miller. A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 22:340–349, 1974.

[12] B. Golden, S. Raghavan, and E. Wasil. Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 487–501. Springer, 2008.

[13] M.-H. Ha, N. Bostel, A. Langevin, and L.-M. Rousseau. An exact algorithm for the close enough traveling salesman problem with arc covering

constraints. In *Proceedings of the 1st International Conference on Operations Research and Enterprise Systems*, Vilamoura, Algarve, Portugal, February 2012.

[14] M. Hachicha, M. J. Hodgson, G. Laporte, and F. Semet. Heuristics for the multi-vehicle covering tour problem. *Computers and Operations Research*, 27:29–42, 2000.

[15] M-J. Hodgson, G. Laporte, and F. Semet. A covering tour model for planning mobile health care facilities in suhum district gana. *Journal of regional science*, 38:621–638, 1998.

[16] N. Jozefowiez. A column generation approach for the multi-vehicle covering tour problem. In *Proceedings of Recherche Opérationnelle et Aide à la Décision Française (ROADEF 2011)*, Saint-Etienne, France, March 2011.

[17] L. Labadi, C. Prins, and M. Reghioui. A memetic algorithm for the vehicle routing problem with time windows. *RAIRO-Operations Research*, 42:415–431, 2008.

[18] M. Labbé and G. Laporte. Maximizing user convenience and postal service efficiency in post box location. *Belgian journal of operations research, statistics and computer science*, 26:21–35, 1986.

[19] A. Langevin, M. Desrochers, J. Desrosiers, S. Gélinas, and F. Soumis. A two-commodity flow formulation for the traveling salesman and the makespan problems with time windows. *Networks*, 23:631–640, 1993.

[20] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33:1058–1073, 1985.

[21] J. Lysgaard, A. N. Letford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100:423–445, 2004.

[22] S. U. Ngueveu, C. Prins, and R. Wolfler Calvo. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers and Operations Research*, 37:1877–1885, 2010.

[23] C. Prins. A GRASP x evolutionary local search hybrid for the vehicle routing problem. In F. B. Pereira and J. Tavares, editors, *Bio-inspired Algorithms for the Vehicle Routing Problem*, Studies in Computational Intelligence, pages 35–53. Springer Verlag, 2009.

[24] C. Prins. Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 22:916–928, 2009.

[25] C. Prins, N. Labadi, and M. Reghiouit. Tour splitting algorithms for vehicle routing problems. *International Journal of Production Research*, 47:507–535, 2008.

[26] M. Sánchez-García, M. I. Sobrón, and B. Vitoriano. On the set covering polytope: Facets with coefficients in {0, 1, 2, 3}. *Annals of Operations Research*, 81:343–356, 1998.

[27] J-C. Simms. Fixed and mobile facilities in dairy practice. *Veterinary clinics of North America - Food animal practice*, 5:591–601, 1989.

[28] W. Swaddiwudhipong, C. Chaovakiratipong, P. Nguntra, P. Lerdlukanavonge, and S. Koonchote. Effect of a mobile unit on changes in knowledge and use of cervical cancel screening among rural thai women. *International journal of epidemiology*, 24:493–498, 1995.

Data	m	Nv	Node	Time	Gap_{BnC}	Result
A1-1-25-75-4	2	8	7	1.13	0	8479
A1-1-25-75-5	2	8	111	3.27	0	8479
A1-1-25-75-6	2	8	441	6.87	0	8479
A1-1-25-75-8	1	8	1827	20.10	0	7985
A1-5-25-75-4	2	8	489	9.49	0	10827
A1-5-25-75-5	2	8	0	0.11	0	8659
A1-5-25-75-6	2	8	6	0.63	0	8659
A1-5-25-75-8	1	8	169	4.20	0	8265
A1-1-50-50-4	3	11	211	9.91	0	10271
A1-1-50-50-5	2	11	249	12.36	0	9220
A1-1-50-50-6	2	11	1223	24.79	0	9130
A1-1-50-50-8	2	11	12370	203.93	0	9130
A1-10-50-50-4	5	19	312215	4828.55	0	17953
A1-10-50-50-5	4	19	10554	173.61	0	15440
A1-10-50-50-6	3	19	123466	1586.21	0	14064
A1-10-50-50-8			357749	7200.12	5.61	
B1-1-25-75-4	2	7	20	1.81	0	7146
B1-1-25-75-5	2	7	87	3.23	0	6901
B1-1-25-75-6	1	7	102	4.33	0	6450
B1-1-25-75-8	1	7	593	10.88	0	6450
B1-5-25-75-4	2	9	0	0.22	0	9465
B1-5-25-75-5	2	9	178	5.74	0	9460
B1-5-25-75-6	2	9	1177	18.07	0	9148
B1-5-25-75-8	1	9	465	8.94	0	8306
B1-1-50-50-4	2	9	306	16.63	0	10107
B1-1-50-50-5	2	9	3642	84.08	0	9723
B1-1-50-50-6	2	10	8941	162.24	0	9382
B1-1-50-50-8	2	10	3862	76.06	0	8348
B1-10-50-50-4	4	17	8560	127.64	0	15209
B1-10-50-50-5	3	16	8504	149.24	0	13535
B1-10-50-50-6	3	16	6115	104.70	0	12067
B1-10-50-50-8	2	17	2000	32.27	0	10344
C1-1-25-75-4	1	5	21	2.82	0	6161
C1-1-25-75-5	1	5	177	5.81	0	6161
C1-1-25-75-6	1	5	289	7.73	0	6161
C1-1-25-75-8	1	5	577	9.42	0	6161
C1-5-25-75-4	2	9	6	0.39	0	9898
C1-5-25-75-5	2	9	112	2.98	0	9707
C1-5-25-75-6	2	9	170	4.17	0	9321
C1-5-25-75-8	2	9	1	0.35	0	7474
C1-1-50-50-4	3	11	258	8.12	0	11372
C1-1-50-50-5	2	10	354	13.28	0	9900
C1-1-50-50-6	2	11	2671	56.91	0	9895
C1-1-50-50-8	2	10	109	8.47	0	8699
C1-10-50-50-4	4	17	9647	164.37	0	18212
C1-10-50-50-5	4	17	8592	126.79	0	16362
C1-10-50-50-6	3	17	15289	240.39	0	14749
C1-10-50-50-8	2	17	303	5.62	0	12394
D1-1-25-75-4	2	7	13	1.04	0	7671
D1-1-25-75-5	2	7	256	5.38	0	7465
D1-1-25-75-6	1	7	89	3.80	0	6651
D1-1-25-75-8	1	7	1037	12.85	0	6651
D1-5-25-75-4	2	9	78	1.65	0	11820
D1-5-25-75-5	2	9	1626	16.72	0	10982
D1-5-25-75-6	2	10	150	3.40	0	9669
D1-5-25-75-8	1	9	129	1.25	0	8200
D1-1-50-50-4	3	10	95	9.34	0	11606
D1-1-50-50-5	2	11	938	29.32	0	10770
D1-1-50-50-6	2	11	12461	281.28	0	10525
D1-1-50-50-8	2	10	5222	110.62	0	9361
D1-10-50-50-4	5	19	393	10.93	0	20982
D1-10-50-50-5	4	18	27584	393.45	0	18576
D1-10-50-50-6	3	17	6574	116.08	0	16330
D1-10-50-50-8	3	17	15423	248.39	0	14204

Table 2: Computational results of branch-and-cut algorithm on instances with $nb_{total} = 100$

Data	m	Nv	Node	Time	Gap_{BnC}	Result
A2-1-50-150-4	2	9	150	82.21	0	11550
A2-1-50-150-5	2	10	1476	340.58	0	10407
A2-1-50-150-6	2	11	6498	1075.80	0	10068
A2-1-50-150-8	1	9	359	153.40	0	8896
A2-10-50-150-4	4	16	7108	1256.98	0	17083
A2-10-50-150-5	3	16	2374	494.66	0	14977
A2-10-50-150-6	3	16	4575	978.92	0	13894
A2-10-50-150-8	2	16	1164	280.21	0	11942
A2-1-100-100-4	3	10	25896	4593.91	0	11885
A2-1-100-100-5	2	11	6079	1440.13	0	10234
A2-1-100-100-6			23889	7200.13	5.85	
A2-1-100-100-8			24359	7200.12	12.88	
A2-20-100-100-4			43723	7200.08	1.97	
A2-20-100-100-5			30722	7200.13	4.38	
A2-20-100-100-6			31371	7200.14	4.43	
A2-20-100-100-8			27579	7200.08	8.76	
B2-1-50-150-4	3	10	253	166.00	0	11175
B2-1-50-150-5	2	10	4429	1114.67	0	10502
B2-1-50-150-6	2	10	6527	1273.97	0	9799
B2-1-50-150-8	2	10	2636	629.77	0	8846
B2-10-50-150-4	5	17	34309	5972.52	0	16667
B2-10-50-150-5	4	17	424	124.21	0	14188
B2-10-50-150-6	3	17	3893	773.56	0	12954
B2-10-50-150-8	2	17	3151	732.65	0	11495
B2-1-100-100-4	4	17	48551	6614.98	0	18370
B2-1-100-100-5	4	17	8181	1471.99	0	15876
B2-1-100-100-6			26450	7200.08	4.65	
B2-1-100-100-8			27427	7200.09	6.60	
B2-20-100-100-4			44254	7200.14	1.00	
B2-20-100-100-5			34898	7200.11	4.24	
B2-20-100-100-6			3536	7200.16	4.99	
B2-20-100-100-8			38336	7200.10	8.97	

Table 3: Computational results of branch-and-cut algorithm on instances with $nb_{total} = 200$

Data	Imp	Go	Do1	Do2	Cov	Flow	Cap	$LB0$	$LB1$	Gap_{LB}
A1-1-25-75-4	24	9	95	3	0	105	0	6018.33	8217.95	36.55
A1-1-25-75-5	32	11	135	66	0	157	0	5473.29	7521.08	37.41
A1-1-25-75-6	37	23	165	127	0	201	0	5100.11	7411.28	45.32
A1-1-25-75-8	34	21	195	161	0	309	0	4075.52	6340.25	55.57
A1-5-25-75-4	36	25	135	90	0	500	0	8052.23	9466.44	17.56
A1-5-25-75-5	8	16	11	1	0	8	0	7544.67	8659.00	14.77
A1-5-25-75-6	8	11	39	22	0	48	0	7243.45	8576.57	18.40
A1-5-25-75-8	25	14	78	76	0	125	0	6107.76	7398.16	21.13
A1-1-50-50-4	80	10	318	88	1	489	0	7395.66	9345.99	26.37
A1-1-50-50-5	98	9	330	160	1	503	0	5990.35	7925.38	32.30
A1-1-50-50-6	101	2	366	374	3	562	0	5449.91	7186.33	31.86
A1-1-50-50-8	121	8	491	617	1	756	0	4741.54	6325.92	33.41
A1-10-50-50-4	115	21	546	457	0	1062	0	14017.99	16223.01	15.73
A1-10-50-50-5	81	31	336	291	0	749	0	11943.33	14317.22	19.88
A1-10-50-50-6	81	40	445	226	2	913	0	10367.16	12769.89	23.18
A1-10-50-50-8	82	22	679	548	2	11382	6	9252.83	11280.34	21.91
B1-1-25-75-4	30	6	106	12	3	145	0	5070.24	6563.78	29.46
B1-1-25-75-5	30	6	124	38	3	173	0	4602.00	5821.71	26.50
B1-1-25-75-6	45	7	138	53	5	183	0	3835.19	5009.56	30.62
B1-1-25-75-8	36	9	201	90	9	296	0	3512.53	4464.54	27.10
B1-5-25-75-4	28	3	26	0	6	52	0	8198.25	9465.00	15.45
B1-5-25-75-5	23	12	94	43	6	180	0	7418.07	8713.15	17.46
B1-5-25-75-6	32	17	153	77	13	297	0	6877.75	8291.14	20.55
B1-5-25-75-8	27	11	95	51	8	220	0	5757.54	7193.69	24.94
B1-1-50-50-4	146	8	369	47	16	705	0	6546.09	8706.10	33.00
B1-1-50-50-5	166	12	501	249	20	977	0	5640.16	7750.26	37.41
B1-1-50-50-6	151	18	580	374	11	985	0	5038.11	7193.29	42.78
B1-1-50-50-8	161	6	532	395	27	779	0	3854.96	5744.45	49.01
B1-10-50-50-4	53	30	264	162	21	778	0	11940.07	14106.86	18.15
B1-10-50-50-5	52	27	314	186	23	802	0	10037.92	12125.23	20.79
B1-10-50-50-6	66	17	298	99	15	699	0	9106.06	10815.94	18.78
B1-10-50-50-8	56	25	218	62	18	516	3	7692.75	9342.99	21.45
C1-1-25-75-4	40	8	106	2	0	131	0	3824.38	5265.87	37.69
C1-1-25-75-5	44	9	133	50	2	177	0	3554.07	4946.48	39.18
C1-1-25-75-6	45	14	137	46	11	185	0	3364.92	4678.16	39.03
C1-1-25-75-8	41	12	157	120	4	588	0	3090.08	4279.73	38.50
C1-5-25-75-4	16	4	36	1	5	49	0	9010.92	9795.45	8.71
C1-5-25-75-5	23	10	87	29	21	125	0	8441.60	9205.33	9.05
C1-5-25-75-6	15	24	107	31	24	151	0	8077.32	8688.03	7.56
C1-5-25-75-8	16	3	43	0	0	63	0	6629.66	7474.00	12.74
C1-1-50-50-4	56	7	231	60	19	384	0	8435.52	10462.34	24.03
C1-1-50-50-5	74	10	315	120	9	360	0	6826.63	8896.96	30.33
C1-1-50-50-6	88	15	375	287	7	579	0	6242.41	8210.56	31.53
C1-1-50-50-8	56	14	234	81	6	294	0	5546.26	7479.82	34.86
C1-10-50-50-4	81	32	337	155	24	720	0	14892.18	17216.99	15.61
C1-10-50-50-5	65	24	283	237	18	590	0	13414.50	15583.38	16.17
C1-10-50-50-6	71	44	331	250	12	655	0	11648.63	13932.40	19.61
C1-10-50-50-8	31	11	102	12	0	209	0	9701.08	12000.56	23.70
D1-1-25-75-4	12	8	77	8	0	98	0	5665.90	7471.37	31.87
D1-1-25-75-5	24	7	109	42	0	197	0	5125.49	6630.58	29.36
D1-1-25-75-6	22	9	100	8	0	170	0	4088.89	5764.07	40.97
D1-1-25-75-8	27	13	172	98	0	235	0	3709.17	5187.67	39.86
D1-5-25-75-4	19	9	87	5	18	160	0	9291.02	11241.78	20.71
D1-5-25-75-5	51	18	171	43	18	298	0	8192.40	9870.06	20.48
D1-5-25-75-6	21	18	89	11	72	178	0	7521.79	8976.05	19.33
D1-5-25-75-8	16	9	66	1	18	119	0	6180.13	7516.38	21.62
D1-1-50-50-4	102	4	358	12	37	672	0	8180.47	10704.44	30.85
D1-1-50-50-5	124	7	430	129	38	747	0	6801.93	9139.56	34.37
D1-1-50-50-6	173	5	648	383	27	1048	0	5992.69	8264.90	37.92
D1-1-50-50-8	154	10	511	365	35	830	0	5013.83	6954.79	38.71
D1-10-50-50-4	50	12	250	45	9	640	5	17996.72	20346.85	13.06
D1-10-50-50-5	55	26	373	139	21	935	15	15024.75	17130.57	14.02
D1-10-50-50-6	59	23	309	136	11	822	0	12999.15	15073.76	20.81
D1-10-50-50-8	77	16	313	188	14	874	6	10775.99	12613.53	17.05

Table 4: Details of branch-and-cut algorithm on instances with $nb_{total} = 100$

Data	Imp	Go	Do1	Do2	Cov	Flow	Cap	$LB0$	$LB1$	Gap_{LB}
A2-1-50-150-4	86	8	235	14	31	407	0	7453.86	10437.83	40.03
A2-1-50-150-5	156	11	337	58	44	575	0	6512.04	9168.45	40.79
A2-1-50-150-6	179	18	454	122	56	712	0	5931.43	8527.62	43.77
A2-1-50-150-8	119	13	285	79	22	403	0	4510.18	7169.25	58.96
A2-10-50-150-4	189	25	283	76	46	677	0	14155.92	16049.60	13.38
A2-10-50-150-5	195	15	240	12	80	540	0	11994.84	13904.95	15.92
A2-10-50-150-6	138	31	274	145	68	588	0	11052.65	12959.76	17.25
A2-10-50-150-8	121	17	206	91	56	398	0	9224.62	11207.41	21.49
A2-1-100-100-4	742	2	1368	462	332	2301	0	7748.61	9675.91	24.87
A2-1-100-100-5	557	3	1121	410	259	1972	0	6355.55	8212.05	29.21
A2-1-100-100-6	619	3	1365	843	332	2578	0	5459.96	7162.93	31.19
A2-1-100-100-8	605	4	1491	1061	280	2657	0	4335.14	5883.03	35.71
A2-20-100-100-4	328	19	720	390	255	2232	150	21689.48	25042.62	15.46
A2-20-100-100-5	267	14	883	523	238	2486	75	18207.18	21331.41	17.16
A2-20-100-100-6	285	12	825	667	224	2245	40	15921.97	18775.14	17.92
A2-20-100-100-8	267	13	1028	767	238	2571	6	13150.13	15552.06	18.27
B2-1-50-150-4	137	6	373	39	95	624	0	6922.26	9572.96	38.29
B2-1-50-150-5	198	8	564	159	213	925	0	5918.43	8245.40	39.32
B2-1-50-150-6	184	24	514	193	94	884	0	5323.13	7604.10	42.85
B2-1-50-150-8	195	18	492	225	164	2643	0	4164.34	6213.36	49.20
B2-10-50-150-4	129	22	381	54	120	938	5	12762.43	15125.63	18.52
B2-10-50-150-5	56	26	158	19	78	339	0	11340.13	13574.24	19.70
B2-10-50-150-6	74	19	245	41	101	526	0	9971.28	11947.58	19.82
B2-10-50-150-8	80	22	245	52	132	503	0	8410.43	10078.25	19.83
B2-1-100-100-4	281	8	1246	847	30	2364	0	13110.87	16748.90	27.75
B2-1-100-100-5	248	8	856	750	15	1685	0	10989.93	14214.52	29.34
B2-1-100-100-6	296	7	1124	937	22	2542	0	9411.66	12394.72	31.70
B2-1-100-100-8	249	7	1207	1101	27	2311	0	7523.01	10366.04	37.79
B2-20-100-100-4	161	28	756	431	58	2241	31	29049.93	32913.34	13.30
B2-20-100-100-5	135	22	927	537	51	2620	96	23939.47	27494.07	14.85
B2-20-100-100-6	144	11	999	569	67	2711	82	20706.77	23954.74	15.69
B2-20-100-100-8	162	18	1090	647	52	3307	41	16794.95	19488.63	16.04

Table 5: Details of branch-and-cut algorithm on instances with $nb_{total} = 200$

Data	LS1	LS2	LS3	LS4	m	Time	Result	Gap_{UB}
A1-1-25-75-4	1675	15188	3010	332	2	0.16	**8479**	0
A1-1-25-75-5	410	18728	612	298	2	0.17	**8479**	0
A1-1-25-75-6	26	119554	1694	314	2	0.16	**8724**	0
A1-1-25-75-8	0	20502	374	319	1	0.16	**7985**	0
A1-5-25-75-4	396	2508	259	24	2	0.13	**10827**	0
A1-5-25-75-5	87	2504	70	20	2	0.14	**8659**	0
A1-5-25-75-6	4	2671	269	20	2	0.16	**8659**	0
A1-5-25-75-8	0	2628	0	19	1	0.14	**8265**	0
A1-1-50-50-4	4097	115543	18088	1161	3	0.80	**10271**	0
A1-1-50-50-5	10343	113686	25981	1141	2	0.78	**9220**	0
A1-1-50-50-6	8789	132666	768	1167	2	0.81	**9130**	0
A1-1-50-50-8	851	124757	4022	1168	1	0.81	**9130**	0
A1-10-50-50-4	19572	352699	91340	790	5	3.26	17973	0.11
A1-10-50-50-5	15487	396159	120965	879	4	3.81	**15440**	0
A1-10-50-50-6	13842	330348	248498	884	3	3.85	**14064**	0
A1-10-50-50-8	2851	454544	15607	816	3	3.92	13369*	5.61
B1-1-25-75-4	612	13666	1592	283	2	0.22	**7146**	0
B1-1-25-75-5	181	16526	2058	313	2	0.18	**6901**	0
B1-1-25-75-6	0	16822	277	311	1	0.23	**6450**	0
B1-1-25-75-8	0	16033	486	315	1	0.20	**6450**	0
B1-5-25-75-4	371	7900	5533	88	2	0.17	**9465**	0
B1-5-25-75-5	560	11756	1079	97	2	0.16	**9460**	0
B1-5-25-75-6	196	12082	822	81	2	0.17	**9148**	0
B1-5-25-75-8	0	14094	221	113	1	0.17	**8306**	0
B1-1-50-50-4	4722	48864	41466	999	2	0.62	**10107**	0
B1-1-50-50-5	4338	80211	5719	1065	2	0.64	**9723**	0
B1-1-50-50-6	1602	69445	4093	838	2	0.58	**9382**	0
B1-1-50-50-8	0	67874	1784	884	2	0.58	**8348**	0
B1-10-50-50-4	20284	203798	220064	763	4	2.53	**15209**	0
B1-10-50-50-5	2583	143178	51826	610	3	2.08	**13535**	0
B1-10-50-50-6	11045	237408	60456	604	3	1.97	**12067**	0
B1-10-50-50-8	10411	218846	119601	603	2	1.99	**10344**	0
C1-1-25-75-4	0	1493	40	75	1	0.16	**6161**	0
C1-1-25-75-5	0	2729	19	119	1	0.16	**6161**	0
C1-1-25-75-6	0	2729	19	119	1	0.15	**6161**	0
C1-1-25-75-8	0	2729	19	119	1	0.17	**6161**	0
C1-5-25-75-4	265	6570	6330	91	2	0.16	**9898**	0
C1-5-25-75-5	560	19909	2131	153	2	0.18	**9707**	0
C1-5-25-75-6	237	18507	1080	143	2	0.19	**9321**	0
C1-5-25-75-8	0	612694	47	122	1	0.19	**7474**	0
C1-1-50-50-4	729	76365	27063	992	3	0.64	**11372**	0
C1-1-50-50-5	2677	84615	18207	1099	2	0.67	**9900**	0
C1-1-50-50-6	1765	96358	1637	1039	2	0.67	**9895**	0
C1-1-50-50-8	180	91985	4164	1007	2	0.65	**8699**	0
C1-10-50-50-4	9962	174488	193441	651	4	2.23	**18212**	0
C1-10-50-50-5	2841	250973	85292	610	4	2.14	**16362**	0
C1-10-50-50-6	8750	272387	9479	698	3	2.00	**14749**	0
C1-10-50-50-8	5417	224571	137632	635	2	2.07	12414	0.16
D1-1-25-75-4	456	9332	1009	197	2	0.16	**7671**	0
D1-1-25-75-5	52	10340	738	178	2	0.16	**7465**	0
D1-1-25-75-6	0	11099	608	187	1	0.15	**6651**	0
D1-1-25-75-8	0	11074	585	212	1	0.16	**6651**	0
D1-5-25-75-4	87	8366	7010	77	2	0.18	**11820**	0
D1-5-25-75-5	460	13077	1443	74	2	0.17	**10982**	0
D1-5-25-75-6	363	14337	1076	85	2	0.17	**9669**	0
D1-5-25-75-8	10	16711	657	77	1	0.17	**8200**	0
D1-1-50-50-4	5405	122525	59874	1236	3	0.93	**11606**	0
D1-1-50-50-5	3834	120039	63496	1134	2	0.85	**10770**	0
D1-1-50-50-6	4361	135090	8384	1070	2	0.82	10680	1.45
D1-1-50-50-8	1563	149689	9205	1170	2	0.93	**9361**	0
D1-10-50-50-4	11357	276458	275638	731	5	3.82	**20982**	0
D1-10-50-50-5	8226	319468	156224	579	4	3.38	**18576**	0
D1-10-50-50-6	4823	276232	185681	450	3	2.91	**16330**	0
D1-10-50-50-8	5627	366309	157389	580	3	3.54	**14204**	0

Table 6: Computational results of metaheuristic on instances with $nb_{total} = 100$

Data	LS1	LS2	LS3	LS4	m	Time	Result	Gap_{UB}
A2-1-50-150-4	874	70356	34886	1024	2	0.89	**11550**	0
A2-1-50-150-5	2922	86127	15372	1014	2	0.87	**10407**	0
A2-1-50-150-6	1486	92947	1102	884	2	0.89	**10068**	0
A2-1-50-150-8	181	90790	5226	1069	1	0.94	**8896**	0
A2-10-50-150-4	9563	180071	126534	513	4	2.03	**17083**	0
A2-10-50-150-5	4457	167133	130439	462	3	1.50	**14977**	0
A2-10-50-150-6	6740	241410	26618	510	3	1.95	**13894**	0
A2-10-50-150-8	6093	234843	71875	566	2	2.07	**11942**	0
A2-1-100-100-4	6558	230334	53549	2791	3	2.89	**11885**	0
A2-1-100-100-5	7759	205319	110844	2839	2	2.82	**10234**	0
A2-1-100-100-6	6289	232113	4360	2533	2	2.92	10020*	5.85
A2-1-100-100-8	1556	253097	10516	2732	2	2.92	9093*	12.88
A2-20-100-100-4	57528	1328048	1656892	1613	7	43.91	26594*	1.97
A2-20-100-100-5	47550	1885843	746084	1417	6	37.29	23419*	4.38
A2-20-100-100-6	51174	2087397	712975	1583	5	39.50	20966*	4.43
A2-20-100-100-8	23543	2314063	295578	1748	4	42.42	18418*	8.76
B2-1-50-150-4	1593	67434	38854	776	3	0.91	**11175**	0
B2-1-50-150-5	5021	88174	15235	833	2	0.90	**10502**	0
B2-1-50-150-6	3080	91896	4423	877	2	0.91	**9799**	0
B2-1-50-150-8	170	80900	3311	766	2	0.87	**8846**	0
B2-10-50-150-4	14351	256685	136664	722	2	2.87	**16667**	0
B2-10-50-150-5	5410	282663	81587	773	2	2.78	**14188**	0
B2-10-50-150-6	9778	256081	35977	591	2	2.53	**12954**	0
B2-10-50-150-8	4927	269622	72082	595	1	2.53	**11495**	0
B2-1-100-100-4	51333	1045925	774566	4908	2	15.03	**18370**	0
B2-1-100-100-5	53439	1274929	614557	5073	2	15.61	**15876**	0
B2-1-100-100-6	28740	1152819	678806	4861	2	14.83	14926*	4.65
B2-1-100-100-8	22866	1348932	475145	4930	1	15.68	13137*	6.60
B2-20-100-100-4	144950	2947716	2554329	2542	9	117.01	34073*	1.00
B2-20-100-100-5	123354	3297460	2981437	2609	7	126.00	29412*	4.24
B2-20-100-100-6	93415	3827705	1968470	2784	6	116.79	25960*	4.99
B2-20-100-100-8	80471	3780534	1248722	2311	5	114.01	22156*	8.97

Table 7: Computational results of metaheuristic on instances with $nb_{total} = 200$

5

Problème généralisé de tournées sur nœuds

Avant de traiter le problème généralisé de tournées sur nœuds (GVRP), nous redéfinissons le problème et présentons les notions utilisées dans ce chapitre.

Soit $G = (V, E)$ un graphe non orienté, où V est l'ensemble des nœuds et E est l'ensemble des arêtes. $V = \{v_0, ..., v_{n-1}\}$ est l'ensemble des n nœuds qui peuvent être visités, le nœud v_0 est le dépôt où m véhicules identiques avec une capacité commune Q sont localisés. $C = \{C_0, ..., C_{K-1}\}$ est un ensemble de K grappes. Chaque grappe C_i excepté C_0, qui contient seulement le dépôt, a une demande D_i et inclut un ensemble de nœuds de V tel que chaque nœud de V appartient à exactement une grappe. Pour chaque $v_i \in V$, $\alpha(i)$ représente la grappe qui contient le nœud v_i. Le nombre de véhicules m peut être fixe ou variable. Un coût c_{ij} est associé avec chaque arrête de $E = \{v_i, v_j : v_i, v_j \in V, i < j\}$. Le GVRP consiste à déterminer m routes pour m véhicules telles que

- Chaque route commence et se termine au dépôt ;
- Exactement un nœud de chaque grappe est visité une seule fois ;
- La demande collectée sur chaque route ne dépasse pas la capacité du véhicule Q ;
- Le coût total est minimisé.

Le GVRP est clairement NP-difficile car il comprend le CVRP si chaque grappe est réduite à un nœud ou à un GTSP si les contraintes de capacité sont relaxées.

Dans le cadre de cette thèse, nous considérons une version du GVRP qui n'a pas été traitée dans la littérature dans laquelle le nombre de véhicules est une variable de décision. Nos contributions sont les suivantes : nous présentons une nouvelle formulation pour le GVRP puis un algorithme de branchement et coupes pour ce problème, ainsi qu'une métaheuristique. Des évaluations numériques montrent que notre approche exacte peut résoudre les instances comprenant jusqu'à 121 nœuds et 51 grappes, et notre métaheuristique donne des solutions de bonne qualité pour les instances testées. Par rapport avec la formulation proposée dans Bektaş et al. [9], notre formulation donne en général les meilleures bornes inférieures et la performance de l'algorithme de branchement et coupes généré est aussi meilleure.

Le chapitre est organisé comme suit : nous proposons successivement une nou-

velle formulation, un algorithme de branchement et coupes et une métaheuristique pour le problème. Ensuite, des évaluations numériques sont rapportées et des conclusions sont présentées. La version provisoire de notre article qui a été soumis à la revue *Computers and Operations Research* vient clôturer ce chapitre.

5.1 Nouvelle formulation pour le GVRP

Nous introduisons à nouveau d'abord la meilleure formulation proposée dans Bektaş et al. [9] pour le GVRP. Notons que Bektaş et al. [9] traitent le problème avec le nombre fixe de véhicules. Pour adapter au cas où le nombre de véhicules est variable, nous considérons simplement le nombre de véhicules m dans la formulation comme une variable de décision. Cette formulation utilise les variables entières $z_{ij}, \{v_i, v_j\} \in E$, qui comptent le nombre de fois l'arrête $\{v_i, v_j\}$ est utilisée. Définissons $\delta(S) = \{(v_i, v_j) \in E : v_i \in S, v_j \notin S\}$. Le GVRP peut être formulé comme suit :

$$\text{Minimiser} \quad \sum_{\{v_i,v_j\}\in E} c_{ij}x_{ij} \tag{5.1}$$

$$\text{sous les contraintes} \quad z\Big(\delta(C_k)\Big) = 2 \quad \forall C_k \in C \setminus C_0 \tag{5.2}$$

$$z\Big(\delta(C_0)\Big) = 2m \tag{5.3}$$

$$z\Big(\delta(S)\Big) + 2 \sum_{\{v_i,v_j\}\in L: i\notin S} z(\{i\} : C_j) \leq 2 \quad \forall C_k \in C \setminus C_0, S \subseteq C_k, L \in \overline{L}_k \tag{5.4}$$

$$\sum_{(v_i\in S_1, v_j\in S_2)} z_{ij} \leq |S| - \left\lceil \frac{D(S)}{Q} \right\rceil \quad \forall S_1 \subseteq S,$$

$$S_2 \subseteq S, S \subseteq C, |S| \geq 2 \tag{5.5}$$

$$z_{ij} = 0, 1, 2 \quad \forall \{v_i, v_j\} \in \delta(0) \tag{5.6}$$

$$z_{ij} = 0, 1 \quad \forall \{v_i, v_j\} \in E \setminus \delta(0) \tag{5.7}$$

$$m \in \mathbb{N}. \tag{5.8}$$

Ici, les contraintes (5.2) assurent que chaque grappe est visitée exactement une fois. Les contraintes (5.3) impliquent que m véhicules quitteront le dépôt. Les contraintes (5.4) sont dénommées les inégalités de même nœud. Elles assurent que si un véhicule arrive à un nœud dans une grappe, il repart du même nœud. Dans ces contraintes, $\overline{L}_k = \{L : L \subseteq \cup_{i\in C_k} L_i, |L \cap L_i| = 1, \forall i \in C_k$ où $L_i = \{i\} \times (C \setminus \{0, \alpha(i)\})$, est défini pour tous les $v_i \in V \setminus v_0$. Les contraintes (5.5) concernent la capacité.

Nous décrivons ensuite une nouvelle formulation de programmation en nombres entiers pour le GVRP. Nous supposons que chaque nœud v_i de V a une demande d_i égale à la demande $D_{\alpha(i)}$ de sa grappe. Autrement dit, tous les nœuds dans une grappe ont la même demande que celle de la grappe. Le graphe original G est d'abord étendu à $\overline{G} = (\overline{V}, \bar{E})$ en ajoutant un nouveau nœud v_n, qui est une copie du dépôt v_0. Nous avons $\overline{V} = V \cup \{v_n\}$, $V' = \overline{V} \setminus \{v_0, v_n\}$, $\overline{E} = E \cup \{(v_i, v_n), v_i \in V'\}$, et $c_{in} = c_{0i} \; \forall \; v_i \in V'$.

Cette formulation demande deux variables de flux, f_{ij} et f_{ji}, pour représenter une arête d'une solution réalisable du GVRP sur laquelle le véhicule transporte une

charge de Q unités. Lorsqu'un véhicule circule de v_i à v_j, le flux f_{ij} représente la charge collectée et le flux f_{ji} représente l'espace vide du véhicule (c'est-à-dire, f_{ji} = Q - f_{ij}).

Soit x_{ij} la variable binaire égale à 1 si l'arête $\{v_i, v_j\}$ est utilisée dans la solution et 0 sinon. Dénotons y_i la variable binaire qui indique la présence du nœud v_i dans la solution. Alors le GVRP peut être décrit comme suit :

$$\text{Minimiser} \quad \sum_{\{v_i,v_j\}\in\overline{E}} c_{ij}x_{ij} \tag{5.9}$$

$$\text{sous les contraintes} \quad \sum_{v_i\in C_k} y_i = 1 \qquad \forall C_k \in C \tag{5.10}$$

$$\sum_{v_i\in\overline{V},i<k} x_{ik} + \sum_{v_j\in\overline{V},j>k} x_{kj} = 2y_k \qquad \forall v_k \in V' \tag{5.11}$$

$$\sum_{v_j\in\overline{V}} (f_{ji} - f_{ij}) = 2d_i y_i \qquad \forall v_k \in V' \tag{5.12}$$

$$\sum_{v_j\in V'} f_{0j} = \sum_{v_i\in V'} d_i y_i \tag{5.13}$$

$$\sum_{j\in V'} f_{nj} = mQ \tag{5.14}$$

$$f_{ij} + f_{ji} = Qx_{ij} \qquad \forall\{v_i, v_j\} \in \overline{E} \tag{5.15}$$

$$f_{ij} \geq 0, f_{ji} \geq 0 \qquad \forall\{v_i, v_j\} \in \overline{E} \tag{5.16}$$

$$x_{ij} = 0, 1 \qquad \forall\{v_i, v_j\} \in \overline{E} \tag{5.17}$$

$$y_i = 0, 1 \qquad \forall v_i \in V' \tag{5.18}$$

$$m \in \mathbb{N}. \tag{5.19}$$

La fonction objectif (5.9) est de minimiser le coût total de parcours. Les contraintes (5.10) assurent que le tour inclut exactement un nœud de chaque grappe, tandis que les contraintes (5.11) assurent que chaque nœud de V' est visité au plus une fois. Les contraintes (5.12)-(5.15) définissent les variables de flux. Spécifiquement, les contraintes (5.12) indiquent que le flux d'entrée moins le flux de sortie à chaque nœud $v_i \in V'$ est égal à $2d_i$ si v_i est utilisé et à 0 sinon. Le flux de sortie au nœud source v_0 (5.13) est égal à la demande totale des nœuds qui sont utilisés dans la solution, et le flux de sortie à v_n (5.14) correspond à la capacité totale de la flotte de véhicules. La contrainte (5.15) est dérivée de la définition des variables de flux. Les contraintes (5.16) et (5.19) définissent les variables.

Le figure 5.1 illustre une solution réalisable du GVRP avec deux routes dans le cas où $Q = 50$ sous la forme de deux marchandises. Les demandes des grappes $C_1, ..., C_5$ sont 10, 20, 20, 20 et 20 respectivement. Les traits pleins dans la figure représentent les flux f_{ij}, tandis que les lignes pointillées représentent les flux f_{ji}.

5.2 Méthodes de résolution proposées

5.2.1 Algorithme de branchement et coupes

Avant de décrire notre algorithme de branchement et coupes, nous présentons les inégalités valides pour le GVRP. Les inégalités suivantes sont déduites à partir

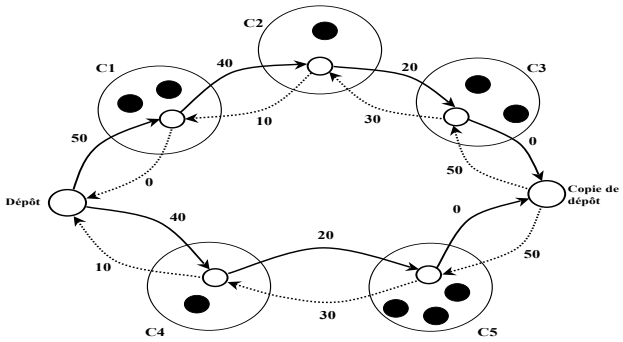

FIGURE 5.1 – Les chemins de flux pour la solution avec deux routes et Q=50

des variables binaires :

$$x_{ij} \leqslant y_i \text{ et } x_{ij} \leqslant y_j \ (v_i \text{ ou } v_j \in V) \tag{5.20}$$

Les inégalités de flux suivantes ont été introduites dans Baldacci et al. [5] :

$$f_{ij} \geq d_j x_{ij}, f_{ji} \geq d_i x_{ij} \text{ si } i, j \neq v_0 \text{ et } i, j \neq v_n. \tag{5.21}$$

Comme cela est montré dans Bektaş et al. [9], chaque instance du GVRP peut être transformée en une instance du CVRP en contractant chaque grappe en un seul nœud. De ce fait, des inégalités valides pour le CVRP comme *peigne, peigne étendu, capacité, capacité généralisée et inégalités de hypotour*, etc. sont aussi valides pour GVRP. Ici, nous limitons aux contraintes de capacité, originalement proposées par Laporte et al. [40] :

$$\sum_{(v_i \in S_1, v_j \in S_2)} x_{ij} \leq |S| - \left\lceil \frac{q(S)}{Q} \right\rceil (S_1 \subseteq S, S_2 \subseteq S, S \subseteq C, |S| \geq 2). \tag{5.22}$$

Nous proposons d'abord de résoudre le GVRP à l'optimalité par un algorithme de branchement et coupes classique. La technique principale est que nous résolvons un programme linéaire contenant les contraintes (5.9), (5.10), (5.11), (5.12), (5.13), (5.14), (5.15), (5.16), $0 \leq x_{ij}, y_i \leq 1$ et $m \geq 0$. Ensuite, nous cherchons les contraintes qui violent des inégalités valides (5.20), (5.21), et (5.22). Les contraintes détectées sont ajoutées au programme actuel, qui est ensuite ré-optimisé. Ce processus est répété jusqu'à ce que toutes les contraintes soient satisfaites. S'il y a des variables fractionnaires, nous branchons pour générer deux nouveaux sous-problèmes. Si toutes les variables sont entières, nous exploitons un autre sous-problème.

La séparation des contraintes de type (5.20) et (5.21) est évidente. Pour générer les contraintes de capacité (5.22), nous utilisons l'algorithme randomisé glouton (*greedy randomized algorithm*), proposé par Augerat et al. [2] et ré-utilisé dans Baldacci et al. [4]. L'implémentation de notre algorithme de branchement et coupes est décrite en détail dans l'article 4 à la fin de ce chapitre.

5.2.2 Une métaheuristique pour le GVRP

Nous introduisons ensuite une métaheuristique qui donne de bonnes solutions pour le GVRP. Elle est utilisée aussi pour fournir des bornes supérieures initiales pour notre algorithme de branchement et coupes. Sa caractéristique nouvelle est une procédure de Découpe qui convertit un tour géant incluant uniquement les grappes (c'est-à-dire que chaque nœud de ce tour est une grappe) et encodé comme les tours de TSP en une solution du GVRP. Cette procédure est ensuite intégrée dans un algorithme hybride entre le GRASP (Greedy Randomized Adaptive Search Procedure) et le ELS (Evolutionary Local Search) proposé dans Prins [49]. Le GRASP peut être considéré comme une recherche locale avec de multiples départs, dans laquelle chaque solution initiale est construite par l'utilisation une heuristique randomisée glouton et améliorée par la recherche locale. Dans la méthode ELS, une seule solution est mutée pour obtenir quelques enfants qui sont ensuite améliorés par la recherche locale. La génération suivante est la meilleure solution entre le parent et ses enfants. Dans ce paragraphe, nous ne décrivons que la caractéristique nouvelle de la métaheuristique - la procédure de Découpe (voir l'article 4 à la fin de ce chapitre pour tous les autres détails).

L'algorithme 1 donne une implémentation efficace pour notre procédure de Découpe. Cette implémentation a été modifiée par rapport à celle que Prins et al. [50] ont utilisé pour décrire la Découpe pour le CVRP.

L'entrée de l'algorithme 1 est une permutation S de K indices des grappes. La procédure de Découpe construit un graphe auxiliaire H avec K nœuds. Chaque sous-séquence de grappes (S_i,...,S_j) qui peut être considérée comme une route (la demande inférieure ou égale à Q) est modélisée par un arc ($i-1$,j) du graphe H. Pour calculer le coût de cet arc, nous utilisons pour chaque nœud $v_k \in S_t$ ($t = i, ..., j$) une étiquette $costsum(v_k)$ représentant le plus court chemin de nœud 0 passant par les grappes $S_i, ..., S_{t-1}$ à v_k. Le coût de l'arc ($i-1$,j) est la valeur la plus petite de $costsum(v_k)$ avec $v_k \in S_j$ et peut être calculé dans $\mathcal{O}(\lambda^2)$ avec λ représentant le nombre maximal de nœuds d'une grappe.

Comme dans Prins et al. [50], l'algorithme 1 implémente la procédure de Découpe sans créer le graphe H explicitement. L'algorithme de Bellman est utilisé pour calculer le plus court chemin de nœud 0 à S_k. Chaque grappe S_k est associée avec une étiquette V_k représentant le coût du plus court chemin de nœud 0 à S_k dans H. Au lieu de stocker chaque arc (i, j) de H, l'étiquette W du nœud j est directement mise à jour s'il y a une amélioration. Soit b le nombre maximal de grappes par route, Prins et al. [50] ont montré que la complexité de l'algorithme de Bellman peut être exprimée en $\mathcal{O}(bK)$. Le coût de chaque arc est calculé en $\mathcal{O}(\lambda^2)$, notre procédure de Découpe se déroule en $\mathcal{O}(bK\lambda^2)$.

5.3 Evaluation numérique

Nos algorithmes sont programmés en C/C++ et exécutés sur un CPU Intel Xeon 2,4 GHz. L'algorithme de branchement et coupes est construit autour de CPLEX 12.4 en utilisant la Callable Library. L'algorithme de pré-traitement proposé dans Bektaş et al. [9] est utilisé dans nos algorithmes pour réduire la taille de problème.

Nous testons nos algorithmes sur les instances de Bektaş et al. [9]. Elles comportent 158 instances issues des instances existantes A, B, P, M et G dans la librairie de CVRP. Les instances de type A, B et P contiennent de 16 à 101 nœuds tandis que

Algorithm 1 Une implémentation de la procédure de Découpe pour le GVRP

$W(0) \Leftarrow 0$;
$P(0) \Leftarrow 0$;
pour $i = 1 \rightarrow K$ **faire**
    $W(i) \Leftarrow +\infty$;
fin pour
pour $i = 1 \rightarrow K$ **faire**
    $j \Leftarrow i$;
    $charge \Leftarrow 0$;
    répéter
        $charge \Leftarrow charge + D_{S_j}$;
        pour tout $v_t \in S_j$ **faire**
            si $i = j$ **alors**
                $costsum(v_t) \Leftarrow 2c_{0v_t}$;
            sinon
                $costsum(v_t) \Leftarrow \min_{v_e \in S_{j-1}} \{costsum(v_e) - c_{0v_e} + c_{v_e v_t} + c_{0v_t}\}$;
            fin si
        fin pour
        si $charge \leq Q$ **alors**
            $W(j) \Leftarrow \min\{W(j); W(i-1) + \min_{v_t \in S_j} costsum(v_t)\}$;
        fin si
        $j \Leftarrow j + 1$;
    jusqu'à $j > K$ ou $charge > Q$
fin pour

les instances de type M et G sont plus grandes avec jusqu'à 262 nœuds. Le nombre de grappes est calculé par $K = \lceil n/\theta \rceil$ en utilisant θ=2 et 3.

Dans le cadre de cette thèse, nous ne présentons que les résultats de la comparaison entre notre formulation avec la meilleure formulation de Bektaş et al. [9] en se basant sur la performance des deux algorithmes de branchement et coupes correspondants. Les résultats de nos algorithmes sont présentés de manière détaillée dans l'article 4 à la fin de ce chapitre. Notons que pour générer les contraintes (5.5), Bektaş et al. [9] ont utilisé les routines heuristiques de Lysgaard [41]. Pour une comparaison équitable, nous utilisons la même procédure pour séparer les contraintes de capacité, c'est-à-dire l'algorithme randomisé glouton proposé par Augerat et al. [2] ainsi que les mêmes bornes supérieures pour l'algorithme de branchement et coupes basé sur la formulation de Bektaş et al. [9]. De plus, tous les paramètres de CPLEX 12.4 sont identiques. Les critères utilisés dans la comparaison sont que le nombre d'instances résolues avec succès (`Succ`), le temps de calcul (Temps), la borne inférieure au nœud de racine ($LB0$), la borne inférieure obtenue à la fin de la résolution (LB) et le nombre de nœuds générés dans l'arbre de branchement et coupes (BB).

Le tableau 5.1 résume les résultats de ces expérimentes. Les résultats en gras indiquent la meilleure formulation entre notre formulation et celle de Bektaş et al. [9]. Set 1 et Set 2 représentent les instances générées en utilisant $\theta = 2$ et 3 respectivement. Les résultats obtenus montrent que notre formulation est meilleure sur presque tous les critères. Nous pouvons résoudre exactement 73 instances sur Set 1, 76 instances sur Set 2, tandis que l'algorithme basé sur la formulation de Bektaş et al. [9] est capable de résoudre 72 instances sur Set 1, 74 instances sur Set 2. Plus précisément, l'algorithme construit sur notre formulation peut résoudre 3 instances supplémentaires : P-n60-k15-C30-V8, A-n63-k9-C21-V3 et A-n80-k10-C27-V4. Il est toutefois un peu plus lent sur deux groupes d'instances (A avec $\theta = 2$ et B avec $\theta = 3$) mais significativement plus rapide sur les autres groupes. Il est aussi plus rapide en terme de temps de calcul en moyenne pour chaque ensemble d'instances (Set 1 et Set 2) et donne de meilleures bornes inférieures $LB0$ ainsi que LB. En terme de bornes inférieures au nœud de racine ($LB0$), notre formulation donne de

meilleurs résultats pour 147 sur 158 instances ; et la formulation de Bektaş et al. [9] est meilleure pour seulement 3 instances. De plus, nous avons besoin de moins de nœuds dans l'arbre de branchement et coupes pour résoudre le problème.

		Ha et al.					Bektaş et al.				
Data	θ	Succ	Temps	$LB0$	LB	BB	Succ	Temps	$LB0$	LB	BB
A	2	26/27	637.67	**615.65**	**633.68**	**1295.1**	26/27	**620.91**	601.10	632.82	2111.0
B	2	23/23	**33.71**	**594.54**	599.30	**120.9**	23/23	48.73	590.74	599.30	530.4
P	2	**23/24**	**541.58**	**348.27**	**357.76**	**1198.7**	22/24	779.83	339.44	357.71	3433.3
M	2	1/4	**16330.31**	**639.62**	**658.66**	**2374.3**	1/4	16813.97	637.32	651.15	5280.8
G	2	0/1	21603.23	**2930.49**	**2945.02**	**233.0**	0/1	21603.67	2809.1	2821.84	328.0
Set 1	2	**73/79**	**1492.59**	**558.79**	**570.37**	**965.2**	72/79	1588.11	548.38	568.12	2190.5
A	3	**27/27**	**356.38**	**453.67**	**471.22**	**1010.9**	25/27	627.66	431.52	468.58	2688.8
B	3	23/23	24.99	**455.42**	459.57	**155.9**	23/23	**15.47**	445.71	459.57	365.9
P	3	24/24	**136.51**	**288.96**	300.94	**414.0**	24/24	191.29	272.75	300.94	1870.5
M	3	2/4	**11918.15**	**495.99**	**508.03**	**4943.5**	2/4	12693.51	479.99	499.65	12010.0
G	3	0/1	21601.39	**2227.79**	**2239.50**	**508.0**	0/1	21601.36	2035.83	2053.25	785.0
Set 2	3	**76/79**	**1037.08**	**417.82**	**428.56**	**742.1**	74/79	1178.75	400.20	424.88	2070.1

Tableau 5.1 – Comparaison avec la méthode de Bektaş et al. [9]

5.4 Conclusion

Dans ce chapitre, nous présentons une nouvelle formulation mathématique, une méthode exacte et une heuristique pour résoudre le problème généralisé de tournées de véhicules (GVRP). La nouvelle formulation est déduite de la formulation de flux de deux marchandises, la méthode exacte est un algorithme de type branchement et coupes classique tandis que la métaheuristique est un algorithme hybride entre GRASP et ELS. Les résultats obtenus sur un ensemble d'instances, comprenant jusqu'à 262 nœuds en tout, et un maximum de 131 nœuds sur le tour, montrent la performance de nos approches. L'algorithme de branchement et coupes basé sur notre formulation est meilleure que celui basé sur la formulation de Bektaş et al. [9] en terme de nombre d'instances résolues, bornes inférieures ainsi que nombre de nœuds requis dans l'arbre de branchement et coupes ; et la métaheuristique donne de bonnes solutions dans un temps de calcul raisonnable.

L'étape suivante de notre recherche est d'appliquer les approches proposées dans ce chapitre pour résoudre d'autres problèmes généralisés, par exemple l'homologue du GVRP défini sur les arcs : le problème généralisé de tournées de véhicules sur arcs (GARP ou Generalized Arc Routing Problem).

Article 4 : An Exact Algorithm and a Metaheuristic for the Generalized Vehicle Routing Problem

version soumise à la revue internationale *Computers and Operations Research*.

An exact algorithm and a metaheuristic for the generalized vehicle routing problem

Minh Hoang Ha[a,c], Nathalie Bostel[b], André Langevin[c], Louis-Martin Rousseau[c,*]

[a] *L'UNAM Université, École des Mines de Nantes, IRCCyN UMR CNRS 6597, 4 rue Alfred Kastler, 44307 Nantes Cedex 3, France*
[b] *L'UNAM Université, Université de Nantes, IRCCyN UMR CNRS 6597, 58 rue Michel Ange, B.P. 420, 44606 Saint-Nazaire Cedex, France*
[c] *Department of Mathematics and Industrial Engineering and CIRRELT, École Polytechnique de Montréal, C.P. 6079, Succursale Centre-ville, Montréal, QC, Canada H3C 3A7*

Abstract

The *generalized vehicle routing problem* (GVRP) involves finding a minimum-length set of vehicle routes passing through a set of clusters, where each cluster contains a number of vertices, such that the tour includes exactly one vertex from each cluster and satisfies capacity constraints. We consider a version of the GVRP where the number of vehicles is a decision variable. This paper introduces a new mathematical formulation based on a two-commodity flow model. We solve the problem using a branch-and-cut algorithm and a metaheuristic that is a hybrid of the *greedy randomized adaptive search procedure* (GRASP) and the *evolutionary local search* (ELS) proposed in [18]. We perform computational experiments on instances from the literature to demonstrate the performance of our algorithms.

Keywords: Generalized vehicle routing, Two-commodity flow model, Branch-and-cut, Metaheuristic

*Corresponding author: Tel.: (+1)514-340-4711 # 4569; fax:(514)340-4463
Email address: louis-martin.rousseau@polymtl.ca (Louis-Martin Rousseau)

1. Introduction

The *capacitated vehicle routing problem* (CVRP) is one of the most popular and challenging combinatorial optimization problems. It involves finding the optimal set of routes for a fleet of vehicles that serves a given set of customers. In classical transportation problems, each customer is served from only one vertex. Therefore, there is always a well-defined set of vertices that must be visited, and we need to find the solution from this set. However, in many real applications a customer can be served from more than one vertex, and the resulting problems are more complex. The GVRP is a generalization of the CVRP and also an extension of the *generalized traveling salesman problem* (GTSP). The GVRP can model problems concerned with the design of bilevel transportation networks; see [6] and [16] for further information on its applications.

The GVRP is defined as follows. Let $G = (V, E)$ be an undirected graph, where V is the vertex set and E is the edge set. $V = \{v_0, ..., v_{n-1}\}$ is the set of n vertices that can be visited, and vertex v_0 is the depot, containing m identical vehicles with a common capacity Q. $C = \{C_0, ..., C_{K-1}\}$ is the set of K clusters. Each cluster C_i except C_0, which contains only the depot, has a demand D_i. Each cluster includes a number of vertices of V, and every vertex in V belongs to exactly one cluster. For each $v_i \in V$, let $\alpha(i)$ be the cluster that contains vertex v_i. The term $D(S) = \sum_{i|C_i \subseteq S} D_i$ is used to represent the total demand in set S which is a subset of V. The number of vehicles m can be constant or variable. A length c_{ij} is associated with each edge of $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$. The GVRP consists in finding m vehicle routes such that (i) each route begins and ends at the depot; (ii) each route visits exactly one vertex of each cluster and visits it only once; (iii) the demand served by each route does not exceed the vehicle capacity Q; and (iv) the total cost is minimized.

The GVRP is clearly NP-hard since it reduces to a VRP when each cluster includes only one vertex or to a GTSP when the capacity constraints are relaxed. The number of papers on this topic is quite limited. The problem was first introduced by Ghiani and Improta [8]. In 2003, Kara and Bektaş [9] proposed the first formulation that was polynomial in the number of constraints and variables. An ant colony algorithm and a genetic algorithm were proposed in [17] and [14] respectively. Recently, Bektaş et al. [6] proposed four formulations and four branch-and-cut algorithms. They concluded that the best formulation was an undirected two-index flow one based

on an exponential number of constraints. They also proposed a heuristic based on *large neighborhood search* (LNS) to provide upper bounds for the branch-and-cut algorithms. At about the same time, Pop et al. [16] introduced two new formulations. The first, the node formulation, is similar to the formulation in [9] but produces a stronger lower bound, and the second is a flow-based formulation. The authors directly solved one instance from [8] using CPLEX. They reported no further computational experience with the proposed formulations and did not develop branch-and-cut algorithms.

In this paper, we consider a version of the GVRP where the number of vehicles is a decision variable. This version has not been investigated in the literature. We make two contributions: i) we present a new formulation for the GVRP, and ii) we propose an exact method and a metaheuristic to solve the problem. Computational experiments show that our exact approach can solve instances of up to 121 vertices and 51 clusters, and our metaheuristic gives high-quality solutions for the instances tested in a reasonable computational time. Compared to the formulation proposed in [6], our formulation provides better lower bounds and better performance of the branch-and-cut algorithm.

The remainder of the paper is organized as follows. Section 2 describes our formulation and several valid inequalities. The branch-and-cut algorithm and the metaheuristic are presented in Sections 3 and 4 respectively. Section 5 discusses the computational results, and Section 6 summarizes our conclusions.

2. New formulation for the GVRP

We first reintroduce the best formulation of [6]. Note that Bektaş et al. [6] tackle the case in which the number of vehicles is constant. To adapt it to the context where the number of vehicles is variable, we simply consider the number of vehicles m in the formulation as a decision variable. This formulation uses integer variables $z_{ij}, (v_i, v_j) \in E$ that count the number of times the edge $\{v_i, v_j\}$ is used. Let $\delta(S) = \{(v_i, v_j) \in E : v_i \in S, v_j \notin S\}$ and $z(F) = \sum_{(v_i,v_j)\in F} z_{ij}$ where F is a subset of E. Then the formulation is as follows:

$$\text{Minimize} \quad \sum_{\{v_i,v_j\}\in E} c_{ij}x_{ij} \tag{1}$$

$$\text{subject to} \quad z\Big(\delta(C_k)\Big) = 2 \quad \forall C_k \in C \setminus C_0 \tag{2}$$

$$z\Big(\delta(C_0)\Big) = 2m \tag{3}$$

$$z\Big(\delta(S)\Big) + 2 \sum_{\{v_i,v_j\}\in L: i\notin S} z(\{i\} : C_j) \leq 2 \quad \forall C_k \in C \setminus C_0, S \subseteq C_k, L \in \overline{L}_k \tag{4}$$

$$\sum_{(v_i\in S_1, v_j\in S_2)} z_{ij} \leq |S| - \left\lceil \frac{D(S)}{Q} \right\rceil \quad \forall S_1 \subseteq S,$$

$$S_2 \subseteq S, S \subseteq C, |S| \geq 2 \tag{5}$$

$$z_{ij} = 0, 1, 2 \quad \forall \{v_i, v_j\} \in \delta(0) \tag{6}$$

$$z_{ij} = 0, 1 \quad \forall \{v_i, v_j\} \in E \setminus \delta(0) \tag{7}$$

$$m \in \mathbb{N}. \tag{8}$$

In this formulation, constraints (2) ensure that each cluster is visited exactly once. Constraints (3) imply that m vehicles will leave the depot. Constraints (4) are referred to as same-vertex inequalities; they ensure that when a vehicle arrives at a certain vertex in a cluster, it will depart from the same vertex. Here, $\overline{L}_k = \{L : L \subseteq \cup_{i\in C_k} L_i, |L \cap L_i| = 1, \forall i \in C_k\}$ where $L_i = \{i\} \times (C \setminus \{0, \alpha(i)\})$, defined for all $v_i \in V \setminus v_0$. Constraints (5) are the capacity constraints.

We now describe a new integer programming formulation for the GVRP. The idea underlying this formulation was first introduced by [7] for the traveling salesman problem (TSP). Langevin et al. [11] extended this approach to the TSP with time windows. Baldacci et al. [3] used it to derive a new formulation and a branch-and-cut algorithm for the VRP, and Baldacci et al. [2] adapted it for the *covering tour problem* (CTP) without capacity constraints. Currently, together with the two-index flow formulation and the set partitioning formulation, this is one of the most successful formulations underlying exact methods for the CVRP (see [4]).

Our formulation is an extension of that proposed by Baldacci et al. [3] for the CVRP. To adapt this idea for the GVRP, we assume that each vertex v_i of V has a demand d_i equal to the demand $D_{\alpha(i)}$ of the cluster to which it belongs. In other words, all the vertices in a cluster have the same demand

as that of the cluster. The difference from CVRP is that we do not need to visit all the vertices of V.

We first extend the original graph G to $\overline{G} = (\overline{V}, \bar{E})$ by adding a new vertex v_n, which is a copy of the depot v_0. We now have $\overline{V} = V \cup \{v_n\}$, $V' = \overline{V} \setminus \{v_0, v_n\}$, $\overline{E} = E \cup \{(v_i, v_n), v_i \in V'\}$, and $c_{in} = c_{0i} \; \forall \; v_i \in V'$.

This formulation requires two flow variables, f_{ij} and f_{ji}, to represent an edge of a feasible GVRP solution along which the vehicle carries a load of Q units. When the vehicle travels from v_i to v_j, flow f_{ij} represents the load collected and flow f_{ji} represents the empty space of the vehicle (i.e., $f_{ji} = Q - f_{ij}$).

Let x_{ij} be a 0-1 variable equal to 1 if edge $\{v_i, v_j\}$ is used in the solution and 0 otherwise. Let y_i be a binary variable that indicates the use of vertex v_i in the solution. Then the GVRP can be stated as:

$$\text{Minimize} \quad \sum_{\{v_i,v_j\}\in\overline{E}} c_{ij}x_{ij} \tag{9}$$

$$\text{subject to} \quad \sum_{v_i\in C_k} y_i = 1 \qquad \forall C_k \in C \tag{10}$$

$$\sum_{v_i\in\overline{V},i<k} x_{ik} + \sum_{v_j\in\overline{V},j>k} x_{kj} = 2y_k \qquad \forall v_k \in V' \tag{11}$$

$$\sum_{v_j\in\overline{V}} (f_{ji} - f_{ij}) = 2d_i y_i \qquad \forall v_k \in V' \tag{12}$$

$$\sum_{v_j\in V'} f_{0j} = \sum_{v_i\in V'} d_i y_i \tag{13}$$

$$\sum_{j\in V'} f_{nj} = mQ \tag{14}$$

$$f_{ij} + f_{ji} = Qx_{ij} \qquad \forall \{v_i, v_j\} \in \overline{E} \tag{15}$$

$$f_{ij} \geq 0, f_{ji} \geq 0 \qquad \forall \{v_i, v_j\} \in \overline{E} \tag{16}$$

$$x_{ij} = 0, 1 \qquad \forall \{v_i, v_j\} \in \overline{E} \tag{17}$$

$$y_i = 0, 1 \qquad \forall v_i \in V' \tag{18}$$

$$m \in \mathbb{N}. \tag{19}$$

The objective (9) is to minimize the total travel cost. Constraints (10) ensure that the tour includes exactly one vertex from each cluster, while

constraints (11) ensure that each vertex of V' is visited at most once. Constraints (12) to (15) define the flow variables. Specifically, constraints (12) state that the inflow minus the outflow at each vertex $v_i \in V'$ is equal to $2d_i$ if v_i is used and 0 otherwise. The outflow at the source vertex v_0 (13) is equal to the total demand of the vertices that are used in the solution, and the outflow at the sink v_n (14) corresponds to the total capacity of the vehicle fleet. Constraint (15) is derived from the definition of the flow variables. Constraints (16) to (19) define the variables.

Figure 1 shows a feasible solution for the GVRP with five clusters and two routes in the case where $Q = 50$ under the two-commodity form. The demands of clusters $C_1, ..., C_5$ are 10, 20, 20, 20, and 20 respectively. The solid lines in the figure represent the flows f_{ij} and the dotted lines represent the flows f_{ji}.

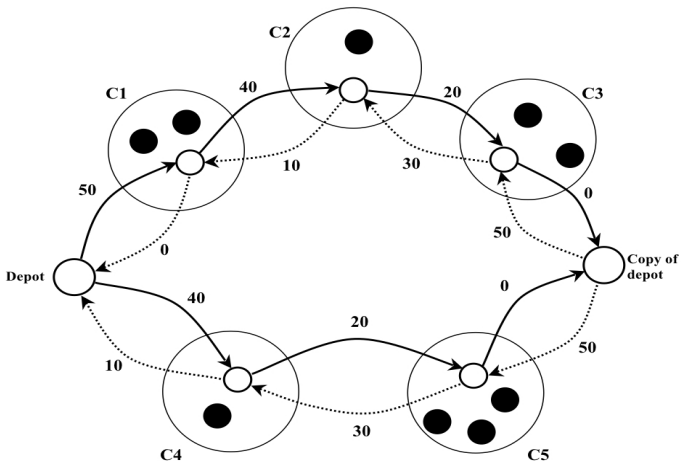

Figure 1: Flow paths for solution with two routes

The linear relaxation of the GVRP can be strengthened by the addition of valid inequalities. The following inequalities are deduced from the definition of the binary variables:

$$x_{ij} \leqslant y_i \text{ and } x_{ij} \leqslant y_j \ (v_i \text{ or } v_j \in V). \tag{20}$$

The following flow inequalities were introduced in [2]:

$$f_{ij} \geq d_j x_{ij}, f_{ji} \geq d_i x_{ij} \text{ if } i, j \neq v_0 \text{ and } i, j \neq v_n. \quad (21)$$

As shown in [6], every GVRP instance can be transformed to a CVRP instance by shrinking each cluster to a single vertex. Therefore, the inequalities that are valid for the CVRP such as the comb, extended comb, capacity, generalized capacity, and hypotour inequalities are also valid for the GVRP. Here, we restrict ourselves to the capacity constraints, originally proposed by [12]:

$$\sum_{(v_i \in S_1, v_j \in S_2)} x_{ij} \leq |S| - \left\lceil \frac{D(S)}{Q} \right\rceil (S_1 \subseteq S, S_2 \subseteq S, S \subseteq C, |S| \geq 2). \quad (22)$$

3. Branch-and-cut algorithm

The GVRP is solved to optimality using a standard branch-and-cut algorithm. We solve a linear program containing the constraints (9), (10), (11), (12), (13), (14), (15), (16), $0 \leq x_{ij}, y_i \leq 1$, and $m \geq 0$. We then search for violated constraints of type (20), (21), and (22), and we add the constraints detected to the current LP, which is then reoptimized. This process is repeated until all the constraints are satisfied. If there are fractional variables, we branch to create two subproblems. If all the variables are integer, we explore another subproblem.

The separation of the constraints of type (20) and (21) is straightforward. To generate the capacity constraints (22), we use the *greedy randomized algorithm*, proposed by [1] and reused in [3].

Our branch-and-cut algorithm is built around CPLEX 12.4 with the Callable Library. As in [6], we enable CPLEX's implementation of strong branching. All the other CPLEX parameters are set to their default values.

We have tested several branching techniques, such as branching on the variables y before x and branching on the variables x before y, but these do not outperform the CPLEX branching. Hence, we let CPLEX make the branching decisions.

4. Metaheuristic

In this section, we present a metaheuristic for the GVRP. It aims to provide good upper bounds for the branch-and-cut algorithm. The main

component is a split procedure that converts a giant tour based on the clusters (i.e., each node of this tour is a cluster) and encoded as a TSP tour into a GVRP solution. This procedure is embedded in an algorithm that is a hybrid of GRASP and the ELS method proposed in [18]. GRASP can be considered as a multi-start local search in which each initial solution is built using a greedy randomized heuristic and then improved by local search. In the ELS method, a single solution is mutated to obtain several children that are then improved by local search. The next generation is the best solution among the parent and its children. We now discuss these procedures in detail.

4.1. Split, concat, and mutate procedures

The split procedure is the backbone of our metaheuristic. It splits a giant tour into GVRP routes. This procedure was originally introduced in [5], and it has been integrated into metaheuristics for various vehicle routing problems (see e.g., [10, 15, 19]). The methods based on the split principle are simple and fast, and they give high-quality solutions. Algorithm 1 gives an efficient implementation of split for the GVRP. This implementation is based on that used in [20] for the split for the CVRP.

The input to Algorithm 1 is a permutation S of the K cluster indices. The split procedure constructs an auxiliary graph H with K nodes. Each subsequence of clusters (S_i,...,S_j) that can be considered as one route (the total demand is not greater than Q) is modeled by an arc ($i-1$,j) of graph H. To compute the cost of this arc, we use for each vertex $v_k \in S_t$ $(t = i, ..., j)$ a label $costsum(v_k)$ representing the shortest path from vertex 0 through clusters $S_i, ..., S_{t-1}$ to v_k. The cost of arc $(i - 1,j)$ is the smallest value of $costsum(v_k)$ for $v_k \in S_j$ and can be computed in $\mathcal{O}(\lambda^2)$ where λ is the maximum number of vertices of a cluster.

As in [20], Algorithm 1 implements the split procedure without creating the graph H explicitly. We use Bellman's algorithm to calculate the shortest path from vertex 0 to S_k. Each cluster S_k is associated with a label V_k representing the cost of the shortest path from vertex 0 to cluster S_k in H. Instead of storing each arc (i, j) of H, we update label W of node j when necessary. Let b be the maximum number of clusters per route. Prins et al. [20] showed that the complexity of Bellman's algorithm is $\mathcal{O}(bK)$. The cost of each arc is computed in $\mathcal{O}(\lambda^2)$, so our split procedure runs in $\mathcal{O}(bK\lambda^2)$.

The concat procedure simply concatenates GVRP routes into a giant tour. It takes the indices of the clusters that include the vertices of the GVRP

Algorithm 1 Split implementation for GVRP

$W(0) \Leftarrow 0$;
$P(0) \Leftarrow 0$;
for $i = 1 \to K$ **do**
    $W(i) \Leftarrow +\infty$;
end for
for $i = 1 \to K$ **do**
    $j \Leftarrow i$;
    $load \Leftarrow 0$;
    repeat
        $load \Leftarrow load + D_{S_j}$;
        for all $v_t \in S_j$ **do**
            if $i = j$ **then**
                $costsum(v_t) \Leftarrow 2c_{0v_t}$;
            else
                $costsum(v_t) \Leftarrow \min_{v_e \in S_{j-1}} \{costsum(v_e) - c_{0v_e} + c_{v_e v_t} + c_{0v_t}\}$;
            end if
        end for
        if $load \leq Q$ **then**
            $W(j) \Leftarrow \min\{W(j); W(i-1) + \min_{v_t \in S_j} costsum(v_t)\}$;
        end if
        $j \Leftarrow j + 1$;
    until $j > K$ or $load > Q$
end for

solution and removes the copies of the depot node. The mutate procedure randomly swaps the position of two vertices in the giant tour.

4.2. Local search procedure

Our local search consists of classical moves: the one-point move, two-point move, 2-opt move, or-opt move, three-point move, and two-adjacent-point move. The one-point move relocates a vertex, and the two-point move swaps the position of two vertices. The three-point move swaps the position of an edge with the position of another vertex. In the 2-opt move, we remove two edges and replace them with two new edges. In the or-opt move, we relocate a string of two vertices. The two-adjacent-point move swaps the position of two edges. All these moves operate both within routes and between routes.

Tests show that the impact of the order of these searches on the performance of our algorithm is not clear. In our implementation, we use the following order to obtain slightly better results: one-point, or-opt, three-point, two-point, 2-opt, and two-adjacent-point.

It is important to note that in the relocation process a vertex can be replaced by vertices in the same cluster. Therefore, the vertices before and after the exchange may be different (see Fig. 2 for an example of a three-point move).

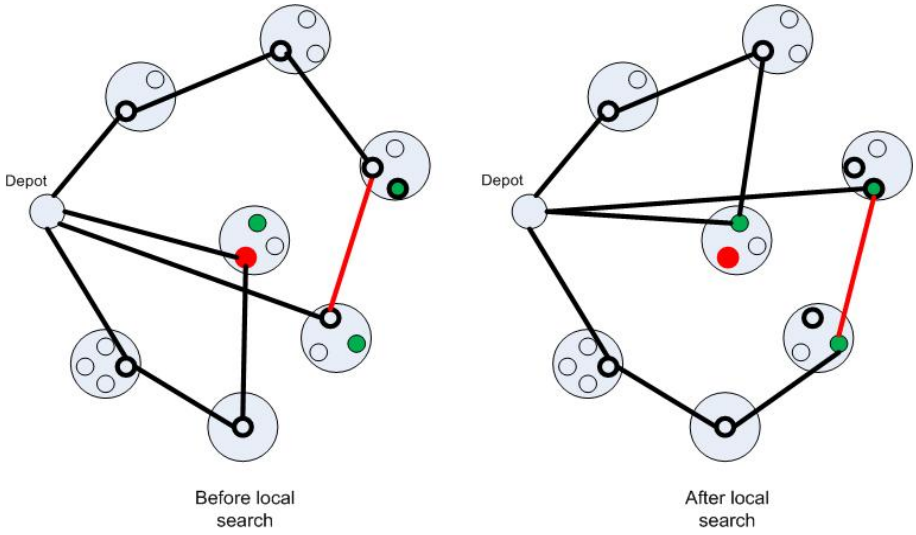

Figure 2: Three-point move for GVRP

4.3. Heuristic for initial solutions

To build an algorithm based on GRASP, we use a randomized version of the nearest-neighbor TSP heuristic to generate a number of GVRP solutions. At each iteration, for a route ending at vertex v_i, we find the nearest uc unrouted clusters. The next vertex of the solution is randomly chosen from the vertices of these uc clusters. Too small a value of uc can limit the algorithm, and too large a value can make the algorithm too random. Our computational tests show that $uc = 2$ gives the best results.

4.4. Metaheuristic algorithm

Algorithm 2 gives the pseudocode for the resulting metaheuristic. S and $f(S)$ denote a solution and its cost respectively. S_{final} and $f(S_{final})$ represent the final results yielded by the metaheuristic. Parameter n_p is the number of phases (each phase generates one final solution for GRASP), n_i the number of iterations per phase, and n_c the number of children generated at each iteration of ELS.

An initial solution S^* is created by the randomized nearest-neighbor heuristic (the RandomInsert procedure) and then improved by the LocalSearch procedure. ELS begins with a giant tour L generated by the concat procedure and performs n_i iterations. At each iteration, we create n_c children of

S by randomly swapping two nodes of the giant tour L (mutate procedure), splitting the giant tour to get a GVRP solution (split procedure), and improving the solution via the LocalSearch procedure. The best child is stored in $\overline{S}$. S^* is updated if the best child is a better solution. The solution S^* is then used for the next ELS iteration.

Algorithm 2 Pseudocode for metaheuristic

```
1: f(S_final) ⇐ +∞;
2: for i = 1 → n_p do
3:     S* ⇐ RandomInsert;
4:     S* ⇐ LocalSearch(S*);
5:     L ⇐ Concat(S*);
6:     for i = 1 → n_i do
7:         f̄ ⇐ +∞;
8:         for j = 1 → n_c do
9:             L ⇐ Mutate(L);
10:            S ⇐ Split(L);
11:            S ⇐ LS(S);
12:            if f(S) < f̄ then
13:                f̄ ⇐ f(S);
14:                S̄ ⇐ S;
15:            end if
16:        end for
17:        if f̄ < f(S*) then
18:            S* ⇐ S̄;
19:        end if
20:    end for
21:    if f(S*) < f(S_final) then
22:        S_final ⇐ S*;
23:        f(S_final) ⇐ f(S*);
24:    end if
25: end for
```

5. Computational experiments

In this section, we reintroduce the GVRP instances and describe the computational evaluation of the proposed algorithms. Our algorithms are coded in C/C++ and the tests are run on a 2.4-GHz Intel Xeon. We use the effective preprocessing algorithm for the GVRP proposed in [6] to reduce the size of the instances.

The parameters n_p, n_i, and n_c in the metaheuristic are chosen empirically. We have tested many combinations, and the following combination gives the best performance in terms of both quality and computational time for our algorithm: $\{n_p, n_i, n_c\}$=$\{30, 80, 20\}$.

We test our algorithms on the Bektaş instances. They were proposed in [6] and include 158 instances derived from the existing instances A, B, P, M,

and G in the CVRP-library. The A, B, and P instances contain 16 to 101 vertices, while the M and G instances are larger with up to 262 vertices. The number of clusters is calculated via $K = \lceil n/\theta \rceil$ with θ=2 and 3.

5.1. Results

This subsection presents the results of our branch-and-cut algorithm and the metaheuristic for the GVRP. The computational time of our exact algorithm is limited, as in [6], to 2 hours for small instances (the Bektaş A, B, and P type instances) and to 6 hours for large instances (the Bektaş M and G type instances). The results are summarized in Table 1, and detailed results can be found in the Appendix. The summary table shows, for each instance, its name (Data), the value of θ (θ), the average computational time for the metaheuristic ($\bar{t}$), and the average computational time for the branch-and-cut algorithm ($\overline{T}$). The $\overline{g}$ column presents the average percentage optimality gap. Let UB be the value of the final solution found by the branch-and-cut algorithm or that found by the metaheuristic (if the branch-and-cut algorithm cannot find a solution). Then the percentage optimality gap g is

$$g = \frac{100(UB - LB)}{UB}. \tag{23}$$

In the next columns, $\overline{BB}$ is the average number of nodes in the branch and bound tree, and $\overline{DO}$, $\overline{FLOW}$, and $\overline{CAP}$ are the average numbers of violated dominance inequalities (20), flow inequalities (21), and capacity inequalities (22) respectively. Finally, `Succ` is the number of instances that were solved to optimality.

The results show that the exact method based on our formulation can solve all instances of type A, B, and P with $\theta = 3$ and of type B with $\theta = 2$. There are two instances of type A and P with $\theta = 2$ for which we cannot find optimal solutions. For the large instances of type M and G, our algorithm can solve two instances with $\theta = 3$ and one with $\theta = 2$. It seems that problems with $\theta = 3$ are easier than those with $\theta = 2$. The problem difficulty increases with n and K. All the types of user cuts are used in the branching process, with the constraints (22) being the most frequently generated (see Tables 6, 7, and 8 in the Appendix).

Our results confirm the high quality of the metaheuristic (see Tables 3, 4, and 5 in the Appendix). The branch-and-cut algorithm provides the optimal

solution for 149 instances, and our metaheuristic finds all these solutions. For the instances where the optimal solution is unknown, the branch-and-cut algorithm cannot improve on the metaheuristic. Moreover, its computational time is acceptable, reaching 861.73 s (about 15 min) for the largest instance, G-n262-k25-C88.

Data	θ	$\overline{t}$	$\overline{T}$	$\overline{g}$	$\overline{BB}$	$\overline{DO}$	$\overline{FLOW}$	$\overline{CAP}$	Succ
A	2	26.54	637.67	0.10	1295.1	184.15	507.26	2367.00	26/27
B	2	28.17	33.71	0.00	120.9	109.39	283.17	1170.70	23/23
P	2	35.18	541.58	0.15	1198.7	155.00	391.63	2350.83	23/24
M	2	279.60	16330.32	2.55	2374.3	669.25	1761.50	25552.25	1/4
G	2	822.84	21610.65	10.84	233.0	911.00	2744.00	38569.00	0/1
A	3	23.55	356.38	0.00	1010.9	200.74	429.93	857.30	27/27
B	3	24.16	25.08	0.00	155.9	111.91	245.91	430.52	23/23
P	3	41.59	102.42	0.00	414.0	169.17	306.46	610.58	24/24
M	3	356.71	11918.15	1.78	4943.5	854.75	1767.00	8975.50	2/4
G	3	861.73	21601.39	9.59	508.0	1243.00	3097.00	22258	0/1

Table 1: Summary of computational results

5.2. Comparison of our formulation with the best one proposed in [6]

We now compare our formulation with the best formulation of [6] by comparing the performance of the branch-and-cut algorithms. To generate the constraints (5), Bektaş et al. [6] used the heuristic routines of [13]. For a fair comparison, we used the same procedure to separate the capacity constraints, i.e., the *greedy randomized algorithm* proposed in [1]. We also used the same upper bounds in the branch-and-cut algorithm based on the formulation of [6]. Moreover, we set the parameters of CPLEX 12.4 to the same values. The criteria used for the comparison are the number of successful instances (`Succ`), the computational time (Time), the lower bound at the root node ($LB0$), the best lower bound at the end of the solution process (LB), and the number of nodes in the branch-and-cut tree (BB).

Table 2 summarizes the results of this experiment, and detailed results can be found in the Appendix (Tables 3, 4, and 5). For each criterion we indicate the better result in bold. Sets 1 and 2 are the instances generated with $\theta = 2$ and 3 respectively. As can be seen, our formulation is better for most of the criteria. We solve one more instance of Set 1 (P-n60-k15-C30-V8) and two more instances of Set 2 (A-n63-k9-C21-V3 and A-n80-k10-C27-V4). It is slightly slower on two groups of instances (A with $\theta = 2$ and B with $\theta = 3$) and significantly faster on the other groups. It is also faster in terms of the average computational time for both Set 1 and Set 2 and gives better values for $LB0$ and LB. Our formulation gives better lower bounds at the

root node ($LB0$) on 147 of the 158 instances; whereas the Bektaş formulation is better on only 3 instances. Moreover, our branch-and-cut trees have fewer nodes.

Data	θ	Ha et al.					Bektaş et al.				
		Succ	Time	$LB0$	LB	BB	Succ	Time	$LB0$	LB	BB
A	2	26/27	637.67	**615.65**	**633.68**	**1295.1**	26/27	**620.91**	601.10	632.82	2111.0
B	2	23/23	**33.71**	**594.54**	599.30	**120.9**	23/23	48.73	590.74	599.30	530.4
P	2	**23/24**	**541.58**	**348.27**	**357.76**	**1198.7**	22/24	779.83	339.44	357.71	3433.3
M	2	1/4	**16330.31**	**639.62**	**658.66**	**2374.3**	1/4	16813.97	637.32	651.15	5280.8
G	2	0/1	21603.23	**2930.49**	**2945.02**	**233.0**	0/1	21603.67	2809.1	2821.84	328.0
Set 1	2	**73/79**	**1492.59**	**558.79**	**570.37**	**965.2**	72/79	1588.11	548.38	568.12	2190.5
A	3	**27/27**	**356.38**	**453.67**	**471.22**	**1010.9**	25/27	627.66	431.52	468.58	2688.8
B	3	23/23	24.99	**455.42**	459.57	**155.9**	23/23	**15.47**	445.71	459.57	365.9
P	3	24/24	**136.51**	**288.96**	300.94	**414.0**	24/24	191.29	272.75	300.94	1870.5
M	3	2/4	**11918.15**	**495.99**	**508.03**	**4943.5**	2/4	12693.51	479.99	499.65	12010.0
G	3	0/1	21601.39	**2227.79**	**2239.50**	**508.0**	0/1	21601.36	2035.83	2053.25	785.0
Set 2	3	**76/79**	**1037.08**	**417.82**	**428.56**	**742.1**	74/79	1178.75	400.20	424.88	2070.1

Table 2: Comparison with method of [6]

6. Conclusion

We have presented a new integer programming formulation, a branch-and-cut algorithm, and a metaheuristic for the GVRP. We have reported computational results for sets of instances from the literature with up to 262 vertices where the tour contains up to 131 vertices. Our results clearly demonstrate the effectiveness of our approach. The branch-and-cut algorithm based on our formulation is more effective than the current state-of-art algorithm of [6] in terms of the number of successful instances, the quality of the lower bounds, and the number of nodes in the branch-and-bound tree. Our metaheuristic gives high-quality solutions in a reasonable computational time.

Future research directions include applying the exact and heuristic approaches proposed in this paper to other problems such as the arc-counterpart of the GVRP, the *generalized arc routing problem*.

References

[1] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a branch-and-cut code for the capacitated vehicle routing problem. Rapport de recherche, ARTEMIS-IMAG, Grenoble, France, 1995.

[2] R. Baldacci, M. A. Boschetti, V. Maniezzo, and M. Zamboni. Scatter search methods for the covering tour problem. In Ramesh Sharda, Stefan Vob, César Rego, and Bahram Alidaee, editors, *Metaheuristic Optimization Via Memory and Evolution*, pages 55–91. Springer Verlag, 2005.

[3] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52:723–738, 2004.

[4] R. Baldacci, A. Mingozzi, and R. Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218:1–6, 2011.

[5] J. E. Beasley. Route first-cluster second methods for vehicle routing. *Omega*, 11:403–408, 1983.

[6] T. Bektaş, G. Erdoğan, and S. Ropke. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, 45(3):299–316, 2011.

[7] G. A. Finke, A. Claus, and E. Gunn. A two-commodity network flow approach to the traveling salesman problem. *Congressus Numerantium*, 41:167–178, 1984.

[8] G. Ghiani and G. Improta. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research*, 122:11–17, 2000.

[9] I. Kara and T. Bektaş. Integer linear programming formulation of the generalized vehicle routing problem. In *Proceedings of the 5th EURO/INFORMS Joint International Meeting*, 2003.

[10] L. Labadi, C. Prins, and M. Reghioui. A memetic algorithm for the vehicle routing problem with time windows. *RAIRO-Operations Research*, 42:415–431, 2008.

[11] A. Langevin, M. Desrochers, J. Desrosiers, S. Gélinas, and F. Soumis. A two-commodity flow formulation for the traveling salesman and the makespan problems with time windows. *Networks*, 23:631–640, 1993.

[12] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33:1058–1073, 1985.

[13] J. Lysgaard. CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. Working paper 03-04, Department of Management Science and Logistics, Aarhus School of Business, Denmark, 2003.

[14] O. Matei, P. C. Pop, and C. Chira. A genetic algorithm for solving the generalized vehicle routing problem. In E.-S. Corchado Rodriguez et al., editor, *Proceedings of HAIS 2010*, volume 6077 of *Lecture Notes in Artificial Intelligence*, pages 119–126, 2010.

[15] S. U. Ngueveu, C. Prins, and R. Wolfler Calvo. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers and Operations Research*, 37:1877–1885, 2010.

[16] P. C. Pop, I. Kara, and A. H. Marc. New mathematical models of the generalized vehicle routing problem and extensions. *Applied Mathematical Modelling*, 36:97–107, 2011.

[17] P. C. Pop, C. M. Pintea, I. Zelina, and D. Dumitrescu. Solving the generalized vehicle routing problem with an ACS-based algorithm. In *Proceedings of BICS 2008*, volume 1117, pages 157–162. American Institute of Physics (AIP), 2009.

[18] C. Prins. A GRASP x evolutionary local search hybrid for the vehicle routing problem. In F. B. Pereira and J. Tavares, editors, *Bio-inspired Algorithms for the Vehicle Routing Problem*, Studies in Computational Intelligence, pages 35–53. Springer Verlag, 2009.

[19] C. Prins. Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 22:916–928, 2009.

[20] C. Prins, N. Labadi, and M. Reghiouit. Tour splitting algorithms for vehicle routing problems. *International Journal of Production Research*, 47:507–535, 2008.

Appendix: Detailed Results of Computational Experiments

In the tables, blank entries indicate that the algorithm did not find a solution. The column headings are as follows:

Data: name of instance;
Ha et al.: branch-and-cut algorithm based on our formulation;
Bektaş et al.: branch-and-cut algorithm based on the formulation of [6];
Node: number of nodes in search tree of branch-and-cut algorithm;
Time: computational time in seconds;
m: number of vehicles in solution;
$LB0$: value of lower bound at root node;
LB: value of best lower bound after branching;
DO: number of constraints of type (20);
$FLOW$: number of constraints of type (21);
CAP: number of constraints of type (22);
SAM: number of constraints of type (4).

Data	Metaheuristic			Ha et al.				Bektaş et al.			
	m	Time	Result	Time	$LB0$	LB	BB	Time	$LB0$	LB	BB
A-n32-k5-C16	3	11.38	508	15.29	481.88	508.00	205	6.72	464.81	508.00	430
A-n33-k5-C17	3	12.43	451	4.07	447.25	451.00	8	0.83	425.25	451.00	67
A-n33-k6-C17	3	10.08	465	2.04	464.64	465.00	7	0.11	454.50	465.00	8
A-n34-k5-C17	3	11.72	489	1.65	489.00	489.00	3	0.22	483.22	489.00	12
A-n36-k5-C18	3	18.05	502	13.92	483.95	502.00	171	3.49	474.83	502.00	178
A-n37-k5-C19	3	14.61	432	0.47	432.00	432.00	3	0.18	425.50	432	6
A-n37-k6-C19	3	12.48	584	10.87	569.77	584.00	100	13.52	548.83	584.00	530
A-n38-k5-C19	3	18.39	476	1.93	471.29	476.00	12	0.60	455.25	476.00	30
A-n39-k5-C20	3	14.89	557	6.91	532.05	557.00	64	15.10	521.43	557.00	605
A-n39-k6-C20	3	16.25	544	2.36	535.88	544.00	21	1.11	525.50	544.00	32
A-n44-k6-C22	3	17.61	608	7.43	595.53	608.00	79	9.16	563.88	608.00	229
A-n45-k6-C23	4	18.80	613	5.60	599.13	613.00	81	4.35	581.75	613.00	174
A-n45-k7-C23	4	23.09	674	373.30	637.25	674.00	2301	275.07	623.20	674.00	3435
A-n46-k7-C23	4	21.45	593	11.65	575.65	593.00	95	4.55	561.50	593.00	101
A-n48-k7-C24	4	20.62	667	197.96	631.04	667.00	1576	111.44	614.92	667.00	2471
A-n53-k7-C27	4	28.11	603	12.93	594.47	603.00	88	5.50	583.06	603.00	60
A-n54-k7-C27	4	31.43	690	51.62	670.97	690.00	243	52.35	655.86	690.00	631
A-n55-k9-C28	5	27.24	699	17.08	683.10	699.00	98	61.60	660.75	699.00	907
A-n60-k9-C30	5	31.98	769	40.83	753.67	769.00	202	30.98	738.21	769.00	298
A-n61-k9-C31	5	34.43	638	50.01	616.52	638.00	216	21.95	620.13	638.00	177
A-n62-k8-C31	4	44.84	740	39.15	717.39	740.00	113	37.44	712.42	740.00	287
A-n63-k10-C32	5	35.88	801	2772.27	767.65	801.00	7109	1840.60	753.17	801.00	8649
A-n63-k9-C32	5	47.81	912	5795.28	876.76	912.00	12872	6406.50	856.77	912	22655
A-n64-k9-C32	5	35.90	763	202.36	743.46	763.00	626	328.81	724.75	763.00	1900
A-n65-k9-C33	5	43.50	682	41.68	657.24	682.00	156	18.90	659.02	682.00	122
A-n69-k9-C35	5	39.60	680	338.26	652.38	680.00	1257	313.33	639.68	680.00	2157
A-n80-k10-C40	5	74.11	997	7200.07	942.54	969.44	7262	7200.90	902.63	946.14	10846
B-n31-k5-C16	3	9.85	441	0.47	441.00	441.00	5	0.07	440.00	441.00	6
B-n34-k5-C17	3	13.09	472	0.36	472.00	472.00	0	0.03	472.00	472.00	0
B-n35-k5-C18	3	14.64	626	0.79	625.30	626.00	9	0.12	623.50	626.00	6
B-n38-k6-C19	3	14.47	451	1.69	449.92	451.00	6	0.30	449.00	451.00	11
B-n39-k5-C20	3	18.76	357	0.81	354.08	357.00	7	0.25	353.50	357.00	12
B-n41-k6-C21	3	15.07	481	2.89	476.30	481.00	9	1.71	469.99	481.00	82
B-n43-k6-C22	3	23.46	483	11.80	472.94	483.00	100	4.54	471.30	483.00	134
B-n44-k7-C22	4	15.44	540	1.11	538.45	540.00	10	1.85	531.63	540	74
B-n45-k5-C23	3	26.53	497	1.37	497.00	497.00	1	0.67	491.83	497.00	13
B-n45-k6-C23	4	20.47	478	40.64	469.37	478.00	468	23.72	464.67	478.00	896
B-n50-k7-C25	4	24.72	449	1.97	449.00	449.00	1	1.37	441.00	449.00	42
B-n50-k8-C25	5	25.23	916	128.90	891.23	916.00	685	752.02	887.31	916.00	9005
B-n51-k7-C26	4	23.71	651	2.52	650.91	651.00	3	0.60	646.75	651.00	9
B-n52-k7-C26	4	24.37	450	0.96	450.00	450.00	3	0.30	446.50	450.00	4
B-n56-k7-C26	4	30.85	486	2.85	483.64	486.00	13	2.57	482.75	486.00	31
B-n57-k7-C29	4	31.09	751	2.32	751.00	751.00	0	4.60	745.91	751.00	95
B-n57-k9-C29	5	28.34	942	7.15	937.84	942.00	22	18.67	923.50	942.00	206
B-n63-k10-C32	5	39.01	816	23.21	802.34	816.00	118	18.70	795.50	816.00	183
B-n64-k9-C32	5	39.67	509	3.20	508.90	509.00	10	1.58	507.00	509.00	8
B-n66-k9-C33	5	45.72	808	20.31	799.19	808.00	48	9.01	798.79	808.00	37
B-n67-k10-C34	5	55.90	673	60.10	662.56	673.00	183	50.13	660.50	673.00	370
B-n68-k9-C34	5	43.95	704	11.23	701.77	704.00	14	7.19	698.33	704.00	42
B-n78-k10-C39	5	63.58	803	448.76	789.64	803.00	1066	220.81	785.79	803.00	933
P-n16-k8-C8	5	0.90	239	0.05	239.00	239.00	0	0.01	239.00	239.00	0
P-n19-k2-C10	2	3.30	147	0.04	147.00	147.00	0	0.00	147.00	147.00	0
P-n20-k2-C10	2	4.25	154	0.05	154.00	154.00	0	0.01	154.00	154.00	1
P-n21-k2-C11	2	4.94	160	0.08	160.00	160.00	1	0.02	156.75	160.00	5
P-n22-k2-C11	2	5.89	162	0.32	162.00	162.00	0	0.03	159.50	162.00	3
P-n22-k8-C11	5	2.40	314	0.71	314.00	314.00	0	0.04	311.50	314.00	6
P-n23-k8-C12	5	2.05	312	1.64	309.76	312.00	8	0.17	299.83	312.00	23
P-n40-k5-C20	3	19.88	294	2.40	286.91	294.00	28	1.34	279.58	294.00	69
P-n45-k5-C23	3	24.80	337	2.25	332.14	337.00	9	1.42	326.40	337.00	36
P-n50-k10-C25	5	21.81	410	108.92	391.58	410.00	647	197.47	373.13	410.00	2796
P-n50-k7-C25	4	31.38	353	13.26	342.84	353.00	102	19.30	335.10	353.00	444
P-n50-k8-C25	5	22.66	372	48.05	356.64	372.00	312	91.86	340.87	372.00	1329
P-n51-k10-C26	6	17.76	427	5.28	419.02	427.00	26	20.64	410.64	427.00	284
P-n55-k10-C28	5	26.61	415	187.84	397.27	415.00	960	406.26	381.50	415.00	4581
P-n55-k15-C28	9	18.10	551	1141.96	524.82	551.00	2650	2378.86	502.41	551.00	10911
P-n55-k7-C28	4	39.95	361	52.69	346.86	361.00	329	91.16	338.60	361.00	1372
P-n55-k8-C28	4	37.84	361	16.49	351.34	361.00	83	18.96	342.00	361.00	320
P-n60-k10-C30	5	35.88	443	7200.05	417.10	437.29	16120	7200.05	401.53	442.00	30780
P-n60-k15-C30	8	24.07	565	3441.63	539.53	565.00	6012	7200.06	517.53	559.13	25032
P-n65-k10-C33	5	39.89	487	340.68	471.59	487.00	756	531.44	454.80	487.00	3079
P-n70-k10-C35	5	48.51	485	126.43	474.24	485.00	307	123.25	462.75	485.00	614
P-n76-k4-C38	2	99.33	383	33.97	376.62	383.00	85	25.56	373.23	383	89
P-n76-k5-C38	3	90.45	405	16.16	397.79	405.00	24	26.76	394.56	405.00	89
P-n101-k4-C51	2	221.56	455	257.03	446.49	455.00	309	381.16	444.22	455.00	537

Table 3: Results for Bektaş instances with $\theta = 2$

	Metaheuristic			Ha et al.				Bektaş et al.			
Data	m	Time	Result	Time	$LB0$	LB	BB	Time	$LB0$	LB	BB
A-n32-k5-C11	2	7.62	386	1.04	386.00	386.00	0	0.07	360.00	386.00	20
A-n33-k5-C11	2	8.62	315	3.55	314.93	315.00	0	0.08	302.00	315.00	14
A-n33-k6-C11	2	5.21	370	1.83	361.72	370.00	7	0.21	346.28	370	25
A-n34-k5-C12	2	9.12	419	3.93	413.22	419.00	9	0.57	389.45	419	76
A-n36-k5-C12	2	10.69	396	3.90	391.84	396.00	9	0.60	358.00	396.00	82
A-n37-k5-C13	2	11.55	347	0.67	347.00	347.00	2	0.12	339.11	347.00	6
A-n37-k6-C13	2	10.52	431	6.53	412.76	431.00	64	5.96	385.11	431.00	404
A-n38-k5-C13	2	12.27	367	2.24	366.03	367.00	2	0.12	352.00	367.00	14
A-n39-k5-C13	2	17.34	364	5.19	351.18	364.00	31	1.93	322.60	364.00	139
A-n39-k6-C13	2	9.18	403	2.97	402.89	403.00	2	0.16	386.46	403.00	5
A-n44-k6-C15	3	18.76	491	17.27	467.02	491.00	155	28.43	438.41	491.00	696
A-n45-k6-C15	3	13.64	474	4.34	464.03	474.00	9	0.77	437.62	474.00	53
A-n45-k7-C15	3	17.15	475	3.63	461.11	475.00	28	1.96	431.00	475.00	100
A-n46-k7-C16	3	19.68	462	10.77	438.01	462.00	106	7.14	416.65	462.00	355
A-n48-k7-C16	3	19.75	451	12.22	428.69	451.00	133	5.29	416.93	451.00	249
A-n53-k7-C18	3	31.21	440	5.00	421.68	440.00	37	3.58	411.73	440.00	134
A-n54-k7-C18	3	33.36	482	7.35	467.00	482.00	50	29.62	432.63	482.00	680
A-n55-k9-C19	3	23.60	473	8.21	461.57	473.00	71	3.30	446.92	473.00	100
A-n60-k9-C20	3	29.94	595	205.72	563.87	595.00	1192	351.78	532.55	595.00	4810
A-n61-k9-C21	4	28.13	473	11.40	452.12	473.00	68	6.83	439.18	473.00	171
A-n62-k8-C21	3	46.02	596	272.94	557.08	596.00	1133	336.91	542.60	596.00	3644
A-n63-k10-C21	4	24.56	593	306.24	560.99	593.00	1847	74.85	543.11	593.00	1383
A-n63-k9-C21	3	31.25	642	1077.89	608.97	642.00	4458	7200.05	570.31	624.33	29945
A-n64-k9-C22	3	44.78	536	30.83	513.20	536.00	137	12.17	511.86	536.00	97
A-n65-k9-C22	3	30.35	500	10.15	479.37	500.00	63	10.22	459.04	500.00	213
A-n69-k9-C23	3	36.56	520	521.66	487.13	520.00	2550	1664.10	461.66	520.00	13408
A-n80-k10-C27	4	85.11	710	7084.87	669.70	710.00	15132	7200.06	617.94	668.30	15774
B-n31-k5-C11	2	9.17	356	0.54	355.55	356.00	6	0.06	353.00	356.00	14
B-n34-k5-C12	2	10.90	369	0.12	369.00	369.00	0	0.02	368.33	369.00	5
B-n35-k5-C12	2	12.46	501	1.19	500.08	501.00	0	0.08	496.50	501.00	11
B-n38-k6-C13	2	13.01	370	2.54	366.91	370.00	9	0.41	360.97	370.00	28
B-n39-k5-C13	2	8.17	280	0.25	280.00	280.00	3	0.03	279.00	280.00	2
B-n41-k6-C14	2	10.90	407	1.25	407.00	407.00	0	0.28	396.00	407.00	19
B-n43-k6-C15	2	17.99	343	1.59	342.33	343.00	3	0.32	338.00	343.00	12
B-n44-k7-C15	3	12.15	395	3.45	388.14	395.00	17	0.93	383.00	395.00	120
B-n45-k5-C15	2	22.52	410	1.80	409.89	410.00	2	0.20	404.00	410.00	17
B-n45-k6-C15	2	17.57	336	2.00	335.05	336.00	8	0.76	328.75	336.00	40
B-n50-k7-C17	3	25.56	393	1.65	392.98	393.00	2	0.27	390.00	393.00	10
B-n50-k8-C17	3	19.36	598	19.80	589.09	598.00	167	14.41	572.75	598.00	735
B-n51-k7-C17	3	18.90	511	2.02	510.37	511.00	2	0.26	506.50	511.00	9
B-n52-k7-C18	3	18.31	359	0.30	359.00	359.00	0	0.04	359.00	359.00	0
B-n56-k7-C19	3	27.55	356	4.13	351.50	356.00	11	15.70	339.00	356.00	807
B-n57-k7-C19	3	27.69	558	1.73	557.84	558.00	1	1.26	554.00	558.00	27
B-n57-k9-C19	3	29.93	681	280.72	665.25	681.00	1816	201.81	657.74	681.00	3762
B-n63-k10-C21	3	44.98	599	5.96	593.93	599.00	20	6.00	584.50	599.00	119
B-n64-k9-C22	4	34.85	452	3.36	449.05	452.00	16	1.58	445.00	452.00	28
B-n66-k9-C22	3	28.40	609	179.18	586.17	609.00	1180	44.74	583.57	609.00	1129
B-n67-k10-C23	4	54.13	558	9.89	548.63	558.00	40	21.09	451.00	558.00	461
B-n68-k9-C23	3	31.58	523	43.00	514.79	523.00	259	41.59	504.50	523.00	1034
B-n78-k10-C26	4	59.67	606	8.32	601.66	606.00	23	3.90	596.20	606.00	26
P-n16-k8-C6	4	1.20	170	0.03	170.00	170.00	0	0.00	170.00	170.00	0
P-n19-k2-C7	1	3.01	111	0.08	111.00	111.00	0	0.01	106.50	111.00	3
P-n20-k2-C7	1	2.55	117	0.49	117.00	117.00	1	0.02	111.90	117.00	6
P-n21-k2-C7	1	3.11	117	0.19	117.00	117.00	0	0.03	115.06	117	4
P-n22-k2-C8	1	3.15	111	0.08	111.00	111.00	0	0.01	111.00	111.00	0
P-n22-k8-C8	4	1.45	249	0.19	244.89	249.00	11	0.05	233.50	249.00	18
P-n23-k8-C8	3	1.66	174	0.11	174.00	174.00	2	0.01	174.00	174.00	0
P-n40-k5-C14	2	18.09	213	3.68	208.92	213.00	9	0.20	203.75	213.00	16
P-n45-k5-C15	2	22.08	238	4.34	225.63	338.00	47	3.56	209.29	238.00	244
P-n50-k10-C17	4	19.40	292	2.03	287.57	292.00	7	1.83	276.17	292.00	49
P-n50-k7-C17	3	23.62	261	3.78	249.97	261.00	27	2.60	237.84	261.00	130
P-n50-k8-C17	3	23.62	262	1.77	258.76	262.00	8	1.39	248.30	262.00	42
P-n51-k10-C17	4	24.59	309	25.61	292.43	309.00	216	36.71	264.38	309.00	1457
P-n55-k10-C19	4	30.75	301	5.94	292.99	301.00	38	8.00	271.18	301.00	219
P-n55-k15-C19	6	16.15	378	8.59	366.22	378.00	41	12.21	343.37	378.00	244
P-n55-k7-C19	3	46.62	271	20.76	255.88	271.00	162	25.64	236.11	271.00	742
P-n55-k8-C19	3	47.46	274	27.64	261.01	274.00	193	23.57	242.20	274.00	683
P-n60-k10-C20	4	34.38	325	41.05	306.24	325.00	296	70.48	287.50	325.00	1870
P-n60-k15-C20	6	23.57	374	213.65	352.20	374.00	868	302.16	333.28	374.00	3692
P-n65-k10-C22	4	43.86	372	41.62	354.30	372.00	199	439.28	328.78	372.00	6155
P-n70-k10-C24	4	57.58	385	514.15	365.59	385.00	1970	1144.44	343.61	385.00	12857
P-n76-k4-C26	2	126.48	309	72.86	294.79	309.00	214	64.03	281.00	309.00	655
P-n76-k5-C26	2	126.82	309	21.39	301.85	309.00	90	59.48	281.00	309.00	541
P-n101-k4-C34	2	294.95	370	1448.16	352.99	370.00	3065	1247.68	347.71	370.00	4073

Table 4: Results for Bektaş instances with $\theta = 3$

Data	θ	Metaheuristic			Ha et al.				Bektaş et al.			
		m	Time	Result	Time	$LB0$	LB	BB	Time	$LB0$	LB	BB
M-n101-k10-C51	2	5	124.23	542	519.56	534.40	542.00	714	2454.36	527.00	542.00	4834
M-n121-k7-C61	2	4	234.26	719	21600.30	650.88	705.84	4653	21600.15	683.60	705.25	10677
M-n151-k12-C76	2	6	305.96	659	21600.26	626.88	634.65	3238	21600.30	612.10	623.28	4226
M-n200-k16-C100	2	9	453.95	789	21601.15	746.30	752.14	892	21600.85	726.58	734.08	1386
G-n262-k25-C131	2	13	822.84	3303	21610.65	2930.49	2945.02	233	21603.67	2809.10	2821.84	328
M-n101-k10-C34	3	4	152.31	458	2831.89	442.82	458.00	7781	5903.39	433.99	458.00	3882
M-n121-k7-C41	3	3	238.23	527	1639.85	513.85	527.00	2001	1669.98	504.18	527.00	3001
M-n151-k12-C51	3	4	434.29	483	21600.29	462.73	474.32	7865	21600.20	438.45	459.06	10689
M-n200-k16-C67	3	6	601.99	605	21600.56	564.57	572.81	2127	21600.46	543.34	554.54	3720
G-n262-k25-C88	3	9	861.73	2477	21601.39	2227.79	2239.50	508	21601.36	2035.83	2053.25	785

Table 5: Results for large Bektaş instances

Data	Ha et al.					Bektaş et al.			
	m	Result	DO	$FLOW$	CAP	m	Result	CAP	SAM
A-n32-k5-C16	3	508	151	401	844	3	508	1228	211
A-n33-k5-C17	3	451	99	230	404	3	451	617	75
A-n33-k6-C17	3	465	44	73	223	3	465	620	31
A-n34-k5-C17	3	489	58	130	269	3	489	636	56
A-n36-k5-C18	3	502	165	417	891	3	502	1109	180
A-n37-k5-C19	3	432	27	49	248	3	432	718	46
A-n37-k6-C19	3	584	135	400	986	3	584	2188	257
A-n38-k5-C19	3	476	103	184	479	3	476	808	83
A-n39-k5-C20	3	557	74	262	969	3	557	2451	270
A-n39-k6-C20	3	544	113	305	910	3	544	1715	166
A-n44-k6-C22	3	608	119	326	1143	3	608	2487	232
A-n45-k6-C23	4	613	77	238	769	4	613	1653	112
A-n45-k7-C23	4	674	228	719	3818	4	674	7249	433
A-n46-k7-C23	4	593	128	306	1042	4	593	2696	133
A-n48-k7-C24	4	667	200	619	2836	4	667	5075	264
A-n53-k7-C27	4	603	155	400	1265	4	603	3094	209
A-n54-k7-C27	4	690	222	581	2057	4	690	4772	357
A-n55-k9-C28	5	699	143	390	1546	5	699	4982	316
A-n60-k9-C30	5	769	181	566	2071	5	769	5400	288
A-n61-k9-C31	5	638	171	407	2034	5	638	5901	216
A-n62-k8-C31	4	740	231	555	2855	4	740	5915	359
A-n63-k10-C32	5	801	402	1157	7363	5	801	13364	634
A-n63-k9-C32	5	912	512	1425	7337	5	912	15093	995
A-n64-k9-C32	5	763	268	795	3470	5	763	8165	564
A-n65-k9-C33	5	682	154	432	2861	5	682	5730	196
A-n69-k9-C35	5	680	217	661	4775	5	680	9105	303
A-n80-k10-C40	-	-	595	1668	10444	-	-	22189	1287
B-n31-k5-C16	3	441	25	56	146	3	441	447	31
B-n34-k5-C17	3	472	25	49	142	3	472	307	22
B-n35-k5-C18	3	626	48	61	150	3	626	527	38
B-n38-k6-C19	3	451	43	120	383	3	451	914	41
B-n39-k5-C20	3	357	39	92	211	3	357	605	35
B-n41-k6-C21	3	481	44	224	724	3	481	1821	114
B-n43-k6-C22	3	483	101	325	1164	3	483	2779	165
B-n44-k7-C22	4	540	65	175	548	4	540	1645	122
B-n45-k5-C23	3	497	49	107	490	3	497	701	70
B-n45-k6-C23	4	478	134	440	1375	4	478	3027	253
B-n50-k7-C25	4	449	49	95	253	4	449	667	60
B-n50-k8-C25	5	916	264	769	1852	5	916	9775	934
B-n51-k7-C26	4	651	36	84	751	4	651	1113	43
B-n52-k7-C26	4	450	29	46	303	4	450	1014	31
B-n56-k7-C26	4	486	80	205	720	4	486	3131	105
B-n57-k7-C29	4	751	54	155	913	4	751	3925	110
B-n57-k9-C29	5	942	202	423	1316	5	942	4794	319
B-n63-k10-C32	5	816	135	373	1450	5	816	3749	194
B-n64-k9-C32	5	509	56	129	729	5	509	1953	72
B-n66-k9-C33	5	808	230	540	3458	5	808	7043	204
B-n67-k10-C34	5	673	175	436	2362	5	673	7995	317
B-n68-k9-C34	5	704	281	690	2196	5	704	5463	218
B-n78-k10-C39	5	803	352	919	5290	5	803	12921	478
P-n16-k8-C8	5	239	10	17	39	5	239	94	12
P-n19-k2-C10	2	147	8	13	29	2	147	55	14
P-n20-k2-C10	2	154	9	17	42	2	154	43	18
P-n21-k2-C11	2	160	9	11	26	2	160	89	22
P-n22-k2-C11	2	162	19	45	61	2	162	119	33
P-n22-k8-C11	5	314	40	88	143	5	314	303	29
P-n23-k8-C12	5	312	21	160	291	5	312	571	45
P-n40-k5-C20	3	294	98	171	411	3	294	935	106
P-n45-k5-C23	3	337	100	193	588	3	337	1123	94
P-n50-k10-C25	5	410	178	460	2502	5	410	5334	377
P-n50-k7-C25	4	353	179	369	938	4	353	3263	317
P-n50-k8-C25	5	372	163	445	2439	5	372	5049	363
P-n51-k10-C26	6	427	111	260	1094	6	427	3660	267
P-n55-k10-C28	5	415	215	519	3141	5	415	6091	400
P-n55-k15-C28	9	551	328	1008	6528	9	551	16484	533
P-n55-k7-C28	4	361	151	356	1603	4	361	4656	292
P-n55-k8-C28	4	361	150	297	1160	4	361	2441	174
P-n60-k10-C30	-	-	472	1327	11477	-	-	19820	796
P-n60-k15-C30	8	565	287	991	10402	-	-	24928	641
P-n65-k10-C33	5	487	251	612	3915	5	487	9052	473
P-n70-k10-C35	5	485	237	508	3020	5	485	8889	328
P-n76-k4-C38	2	383	182	417	1709	2	238	4507	284
P-n76-k5-C38	3	405	183	380	1372	3	405	4893	244
P-n101-k4-C51	2	455	319	735	3490	2	455	9470	586

Table 6: Details of branch-and-cut algorithms for Bektaş instances with $\theta = 2$

Data	Ha et al.					Bektaş et al.			
	m	Result	DO	$FLOW$	CAP	m	Result	CAP	SAM
A-n32-k5-C11	2	386	61	60	168	2	386	195	43
A-n33-k5-C11	2	315	92	151	178	2	315	188	43
A-n33-k6-C11	2	370	59	120	313	2	370	351	68
A-n34-k5-C12	2	419	83	208	209	2	419	350	86
A-n36-k5-C12	2	396	126	237	146	2	396	187	114
A-n37-k5-C13	2	347	45	71	125	2	347	234	65
A-n37-k6-C13	2	431	134	331	381	2	431	679	306
A-n38-k5-C13	2	367	45	70	162	2	367	284	51
A-n39-k5-C13	2	364	154	269	436	2	364	461	220
A-n39-k6-C13	2	403	93	144	269	2	403	449	67
A-n44-k6-C15	3	491	199	429	862	3	491	1096	659
A-n45-k6-C15	3	474	112	131	465	3	474	424	116
A-n45-k7-C15	3	475	113	268	344	3	475	685	198
A-n46-k7-C16	3	462	168	327	453	3	462	819	283
A-n48-k7-C16	3	451	176	387	698	3	451	860	254
A-n53-k7-C18	3	440	123	273	482	3	440	836	218
A-n54-k7-C18	3	482	151	333	581	3	482	1195	541
A-n55-k9-C19	3	473	128	283	666	3	473	1364	168
A-n60-k9-C20	3	595	336	813	1601	3	595	3054	1016
A-n61-k9-C21	4	473	144	292	760	4	473	1429	213
A-n62-k8-C21	3	596	382	892	1226	3	596	2637	823
A-n63-k10-C21	4	593	347	781	1921	4	593	2715	456
A-n63-k9-C21	3	642	486	1238	2700	3	642	5855	2114
A-n64-k9-C22	3	536	253	546	938	3	536	2415	739
A-n65-k9-C22	3	500	181	369	741	3	500	1732	214
A-n69-k9-C23	3	520	383	848	2326	3	520	6151	948
A-n80-k10-C27	4	710	846	1737	3996	-	-	6418	3198
B-n31-k5-C11	2	356	50	76	68	2	356	175	42
B-n34-k5-C12	2	369	23	27	63	2	369	97	23
B-n35-k5-C12	2	501	47	68	63	2	501	111	49
B-n38-k6-C13	2	370	75	153	288	2	370	515	130
B-n39-k5-C13	2	280	21	32	60	2	280	192	29
B-n41-k6-C14	2	407	47	80	143	2	407	342	104
B-n43-k6-C15	2	343	71	108	110	2	343	393	75
B-n44-k7-C15	3	395	75	186	300	3	395	459	90
B-n45-k5-C15	2	410	63	73	124	2	410	219	49
B-n45-k6-C15	2	336	79	169	228	2	336	583	138
B-n50-k7-C17	3	393	74	88	297	3	393	468	53
B-n50-k8-C17	3	598	190	494	613	3	598	1299	369
B-n51-k7-C17	3	511	55	96	267	3	511	439	64
B-n52-k7-C18	3	359	35	38	205	3	359	450	27
B-n56-k7-C19	3	356	121	316	219	3	356	703	327
B-n57-k7-C19	3	558	52	90	315	3	558	710	200
B-n57-k9-C19	3	681	321	787	911	3	681	2077	698
B-n63-k10-C21	3	599	227	431	792	3	599	1206	205
B-n64-k9-C22	4	452	106	217	240	4	452	1033	118
B-n66-k9-C22	3	609	284	768	2014	3	609	3380	383
B-n67-k10-C23	4	558	167	308	379	4	558	1338	233
B-n68-k9-C23	3	523	212	703	1171	3	523	1772	298
B-n78-k10-C26	4	606	179	348	1032	4	606	2084	149
P-n16-k8-C6	4	170	4	14	26	4	170	60	6
P-n19-k2-C7	1	111	13	15	20	1	111	23	18
P-n20-k2-C7	1	117	21	44	27	1	117	44	35
P-n21-k2-C7	1	117	23	26	19	1	117	83	46
P-n22-k2-C8	1	111	14	25	20	1	111	41	18
P-n22-k8-C8	4	249	42	135	87	4	249	184	32
P-n23-k8-C8	3	174	10	22	41	3	174	100	13
P-n40-k5-C14	2	213	101	146	231	2	213	403	75
P-n45-k5-C15	2	238	171	292	421	2	238	686	224
P-n50-k10-C17	4	292	128	195	403	4	292	907	196
P-n50-k7-C17	3	261	173	260	313	3	261	660	179
P-n50-k8-C17	3	262	123	184	387	3	262	872	175
P-n51-k10-C17	4	309	189	372	659	4	309	1468	350
P-n55-k10-C19	4	301	194	285	554	4	301	1275	237
P-n55-k15-C19	6	378	180	345	1044	6	378	2135	272
P-n55-k7-C19	3	271	195	342	642	3	271	1125	340
P-n55-k8-C19	3	274	196	356	652	3	274	1125	358
P-n60-k10-C20	4	325	253	457	1087	4	325	1999	403
P-n60-k15-C20	6	374	307	699	1601	6	374	3965	735
P-n65-k10-C22	4	372	269	446	875	4	372	3303	649
P-n70-k10-C24	4	385	406	788	1658	4	385	3619	977
P-n76-k4-C26	2	309	228	368	1167	2	309	1693	467
P-n76-k5-C26	2	309	190	287	860	2	309	2576	484
P-n101-k4-C34	2	370	630	1252	1860	2	370	3265	1385

Table 7: Details of branch-and-cut algorithms for Bektaş instances with $\theta = 3$

Data	θ	Ha et al.					Bektaş et al.			
		m	Result	DO	$FLOW$	CAP	m	Result	CAP	SAM
M-n101-k10-C51	2	5	542	334	761	4569	5	542	15370	664
M-n121-k7-C61	2	-	-	756	2143	25240	-	-	59265	1510
M-n151-k12-C76	2	-	-	801	1863	30410	-	-	79106	1659
M-n200-k16-C100	2	-	-	786	2324	41990	-	-	129218	1774
G-n262-k25-C131	2	-	-	911	2744	38569	-	-	286211	2236
M-n101-k10-C34	3	4	458	563	1268	1921	4	458	3882	1407
M-n121-k7-C41	3	3	527	693	1473	7741	3	527	15872	1145
M-n151-k12-C51	3	-	-	992	2008	12557	-	-	22870	2373
M-n200-k16-C67	3	-	-	1171	2319	13683	-	-	39806	3141
G-n262-k25-C88	3	-	-	1243	3097	22258	-	-	93227	2475

Table 8: Details of branch-and-cut algorithms for large Bektaş instances

6

Conclusion générale

Nous nous sommes intéressés dans ce mémoire aux problèmes généralisés de tournées de véhicules. Très peu de travaux dans la littérature ont traité ces problèmes et pourtant ils permettent de modéliser plusieurs applications réelles comme le routage du relevé des compteurs intelligents, le routage des navires dans le transport maritime, le problème de localisation des boîtes postales et la conception des routes pour les équipes de soins mobiles dans les pays en voie de développement, etc.

La revue de la littérature que nous avons réalisée sur ces problèmes nous a permis de concentrer nos recherches sur le CEARP (ou Close Enough Arc Routing Problem), le mCTP (ou multi-vehicle Covering Tour Problem) et le GVRP (ou Generalized Vehicle Routing Problem). En effet ce sont les problèmes les plus généraux ou les problèmes les moins étudiés dans la littérature.

La contribution la plus marquante de cette thèse porte sur les méthodes exactes proposées pour la résolution des problèmes étudiés, qui sont au moins équivalentes aux meilleurs algorithmes de résolution actuellement disponibles. Ce sont des algorithmes de branchement et coupes basés sur des formulations en programmation linéaire en nombres entiers. Plus précisément, pour le CEARP, nous avons proposé deux nouvelles formulations et les avons comparées avec une autre formulation existante dans la littérature. Pour le mCTP et le GVRP, nous avons adapté la formulation connue de flux de deux marchandises pour résoudre avec succès ces problèmes par les algorithmes de branchement et coupes correspondants. Ces algorithmes sont devenus les meilleures méthodes exactes pour ces problèmes jusqu'au moment où cette thèse est publiée.

La deuxième contribution de cette thèse concerne les méthodes approchées qui donnent de bons résultats dans un temps raisonnable. Ce sont des métaheuristiques basées sur le principe classique de la recherche locale évolutive (ELS) et le GRASP. Le point le plus remarquable dans cette contribution est la procédure *Découpe* proposée pour le GVRP. Cette procédure possède une caractéristique intéressante : pour chaque séquence de grappes comme données d'entrée, elle donne en sortie une solution optimale pour le problème, indépendemment des nœuds présents dans les grappes.

L'ensemble de ces recherches ont déjà donné lieux à un article accepté dans la

conférence internationale (*ICORES 2012*), trois articles en revues internationales : deux articles acceptés pour publication (l'un sur *Networks* et l'autre sur *European Journal of Operational Research*) et un article en cours de révision (*Computers and Operations Research*).

Compte tenu de l'intérêt des résultats obtenus, nous souhaitons poursuivre cette recherche dans plusieurs directions :

- Améliorer les algorithmes proposés dans cette thèse. Par exemple, pour les algorithmes de branchement et coupes, l'amélioration peut être réalisée par l'ajout des coupes efficaces spécialisées pour chaque problème.
- Résoudre d'autres problèmes généralisés qui n'ont pas été traités dans la littérature comme mGTSP, mCEARP ou mGARP en adaptant les modèles proposés dans cette thèse ou en développant de nouvelles méthodes.
- Prendre en compte d'autres contraintes dans la résolution des problèmes généralisés comme celles relatives aux fenêtres de temps, aux flottes hétérogènes et aux multiples dépôts, etc.
- Résoudre les problèmes généralisés de tournées sur arcs définis sur les graphes non-orientés et mixtes.
- Adapter nos méthodes pour résoudre d'autres classes de problèmes d'optimisation dans lesquels juste un sous-ensemble de nœuds dans le graphe est visité, par exemple pour les problèmes de tournées avec gains.

Bibliographie

[1] J. Aráoz, E. Fernández, and C. Franquesa. The generalized arc routing problem. In *Proceedings of ROUTE 2011*, 2011. 9

[2] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem. Rapport de recherche, ARTEMIS-IMAG, Grenoble, France, 1995. 104, 106

[3] E. Balas and S. M. Ng. On the set covering polytope : All the facets with coefficients in {0, 1, 2}. *Mathematical Programming*, 43(1–3) :57–69, 1986. 68

[4] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52 :723–738, 2004. 104

[5] R. Baldacci, M. A. Boschetti, V. Maniezzo, and M. Zamboni. Scatter search methods for the covering tour problem. In Ramesh Sharda, Stefan Vob, Crésar Rego, and Bahram Alidaee, editors, *Metaheuristic optimization via memory and evolution*, pages 55–91. Springer Verlag, 2005. 7, 68, 104

[6] R. Baldacci, E. Bartolini, and G. Laporte. Some applications of the generalized vehicle routing problem. *Journal of the Operational Research Society*, 61 : 1072–1077, 2010. 6

[7] M. O. Ball and M. J. Magazine. Sequencing of insertions in printed circuit board assembly. *Operations Research*, 36(2) :192–201, 1988. 15

[8] J. E. Beasley. Route first-cluster second methods for vehicle routing. *Omega*, 11 :403–408, 1983. 7, 69

[9] T. Bektaş, G. Erdoğan, and S. Ropke. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, 45(3) :299–316, 2011. vii, 5, 6, 101, 102, 104, 105, 106, 107

[10] D. Ben-Arieh, G. Gutin, M. Penn, A. Yeo, and A. Zverovitch. Transformations of generalized atsp into atsp. *Operations Research Letters*, 31 :357–365, 2003. 4

[11] E. Benavent, A. Carrotta, A. Corberán, J. M. Sanchis, and D. Vigo. Lower bounds and heuristics for the windy rural postman problem. *European Journal of Operations Research*, (2) :855–869, 2007. 16

[12] B. Bontoux, C. Artigues, and D. Feillet. A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem. *Computers and Operations Research*, 37 :1844–1852, 2010. 4, 5

[13] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12 :568–581, 1964. 7

[14] A. Corberán and C. Prins. Recent results on arc routing problems : An annotated bibliography. *Networks*, 56 :50–69, 2010. 8

[15] A. Corberán, A. N. Letchford, and J. M. Sanchis. A cutting plane algorithm for the general routing problem. *Mathematical Programming*, 90 :291–316, 2001. 16

[16] A. Corberán, E. Mota, and J. M. Sanchis. A comparison of two different formulations for arc routing problems on mixed graphs. *Computers and Operations Research*, 33(12) :3384–3402, 2006. 16

[17] J. Current. *Multiobjective design of transportation networks*. PhD dissertation, Johns Hopkins University, 1981. 7

[18] J. Dong, N. Yang, and M. Chen. Heuristic approaches for a tsp variant : The automatic meter reading shortest tour problem. In E. Baker, A. Joseph, A. Medrotra, and M. Trick, editors, *Extending the Horizons : Advances in Computing, Optimisation, and Decision Technologies*, pages 145–163. Springer Verlag, 2007. 8

[19] M. Drexl. *On some generalized routing problems*. Ph.d. dissertation, Rheinisch-Westfalishe Technische Hochschule Aachen, 2003. 9, 11, 12, 13, 22

[20] M. Dror, editor. *Arc routing : Theory, solutions and applications*. 2000. 8

[21] M. Fischetti, J-J. Salazar-González, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3) :378–394, 1997. 4

[22] N. Garg and G. Konjevod. A polylogarithmic approximation algorithm for the group steiner tree problem. *Journal of Algorithms*, 37 :66–84, 2000. 4

[23] M. Gendreau, G. Laporte, and F. Semet. The covering tour problem. *Operations Research*, 45 :568–576, 1997. 7, 67, 68, 70

[24] G. Ghiani and G. Laporte. A branch-and-cut algorithm for the undirected rural postman problem. *Mathematical Programming*, 87(3) :467–481, 2000. 5, 6, 16

[25] B. Gillett and L. Miller. A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 22 :340–349, 1974. 7

[26] B. Golden, S. Raghavan, and E. Wasil. Advances in meter reading : Heuristic solution of the close enough traveling salesman problem over a street network. In *The Vehicle Routing Problem : Lastest Avances and New Challenges*, pages 487–501. Springer, 2008. 9, 10

[27] D. Gulczynski, J. Heath, and C. Price. The close enough traveling salesman problem : A discussion of several heuristics. In F. Alt, M. Fu, and B. Golden, editors, *Perspectives in Operations Research : Papers in Honor of Saul GassâĂŹ 80th Birthday*, pages 271–283. Springer Verlag, 2006. 8

[28] G. Gutin and D. Karapetyan. A memetic algorithm for the generalized traveling salesman problem. *Natural Computing*, 9 :47–60, 2010. 4, 5

[29] G. Gutin, D. Karapetyan, and N. Krasnogor. A memetic algorithm for the asymmetric generalized traveling salesman problem. In M. Pavone, G. Nicosia, and D. Krasnogor, editors, *Proceedings of Nature Inspired Cooperative Strategies for Optimization (NICSO 2007)*, pages 199–210, 2008. 5

[30] M. Hachicha, M. J. Hodgson, G. Laporte, and F. Semet. Heuristics for the multi-vehicle covering tour problem. *Computers and Operations Research*, 27(1) :29–42, 2000. 7

[31] A-L. Henry-Labordere. The record balacing problem : a dynamic programming solution of a generalized traveling salesman problem. *RAIRO*, pages 43–49, 1969. 4

[32] M-J. Hodgson, G. Laporte, and F. Semet. A covering tour model for planning mobile health care facilities in suhum district, gana. *Journal of regional science*, 38 :621–638, 1998. 7

[33] B. Hu and G-R. Raidl. Effective neighborhood structures for the generalized traveling salesman problem. In J. Van Hemert and C. Cotta, editors, *Proceedings of EvoCOP 2008*, pages 36–47, 2008. 4

[34] H. Huang, X. Yang, Z. Hao, C. Wu, Y. Liang, and X. Zhao. Hybrid chromosome genetic algorithm for generalized traveling salesman problems. In L. Wang, K. Chen, and Y-S. Ong, editors, *Proceedings of ICNC 2005*, pages 137–140, 2005. 5

[35] N. Jozefowiez. A column generation approach for the multi-vehicle covering tour problem. In *Proceedings of Recherche Opérationnelle et Aide à la Décision Française (ROADEF 2011)*, Saint-Etienne, France, March 2011. 7, 65, 69, 70, 71, 72

[36] I. Kara and T. Bektaş. Integer linear programming formulation of the generalized vehicle routing problem. In *Proceedings of the 5th EURO/INFORMS Joint International Meeting*, 2003. 5, 6

[37] D. Karapetyan and G. Gutin. Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem. *European Journal of Operational Research*, 219 :234–251, 2012. 4

[38] M. Labbé and G. Laporte. Maximizing user convenience and postal service efficiency in post box location. *Belgian journal of operations research, statistics and computer science*, 26 :21–35, 1986. 7

[39] G. Laporte and F. Semet. Computational evaluation of a transformation procedure for the symmetric generalized traveling salesman problem. *INFOR*, 37 : 114–120, 1999. 4

[40] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33 :1058–1073, 1985. 104

[41] J. Lysgaard. Cvrpsep : A package of separation routines for the capacitated vehicle routing problem. Working paper 03-04, Department of Management Science and Logistics, Aarhus School of Business, Denmark, 2003. 106

[42] O. Matei, P-C. Pop, and C. Chira. A genetic algorithm for sollving the generalized vehicle routing problem. In E-S. Corchado Rodriguez et al., editor, *Proceedings of HAIS 2010*, volume 6077 of *Lecture Notes in Artificial Intelligence*, pages 119–126, 2010. 5

[43] W-K. Mennell. *Heuristics for solving three routing problems : close-enough traveling salesman problem, close-enough vehicule routing problem, and sequence-dependent team orienteering problem*. Ph.d. dissertation, University of Maryland, 2009. 8

[44] C-E. Noon and J-C. Bean. A lagrangian based approach for the asymmetric generalized traveling salesman problem. *Operations Research*, 39 :623–632, 1991. 4

[45] C-E. Noon and J-C. Bean. An efficient transformation of the generalized traveling salesman problem. *INFOR*, 31 :39–44, 1993. 4

[46] C-M. Pintea, P-C. Pop, and C. Chira. The generalized traveling salesman problem solved with ant algorithms. *Jounal of Universal Computer Science*, 13 :1065–1075, 2007. 5

[47] P-C. Pop, C-M. Pintea, I. Zelina, and D. Dumitrescu. Solving the generalized vehicle routing problem with an ACS-based algorithm. In *Proceedings of BICS 2008*, volume 1117, pages 157–162. American Institute of Physics (AIP), 2009. 5

[48] P-C. Pop, I. Kara, and A-H. Marc. New mathematical models of the generalized vehicle routing problem and extensions. *Applied Mathematical Modelling*, 36 :97–107, 2011. 6

[49] C. Prins. A GRASP x evolutionary local search hybrid for the vehicle routing problem. In F. B. Pereira and J. Tavares, editors, *Bio-inspired Algorithms for the Vehicle Routing Problem*, Studies in Computational Intelligence, pages 35–53. Springer Verlag, 2009. 68, 69, 105

[50] C. Prins, N. Labadi, and M. Reghiouit. Tour splitting algorithms for vehicle routing problems. *International Journal of Production Research*, 47 :507–535, 2008. 105

[51] J. Renaud and F-F. Boctor. An efficient composite heuristic for the symmetric generalized traveling salesman problem. *European Journal of Operational Research*, 108 :571–584, 1998. 4

[52] J. Renaud, F-F. Boctor, and G. Laporte. A fast composite heuristic for the the symmetric traveling salesman problem. *INFORMS Journal on Computing*, 8 : 134–143, 1996. 4

[53] M. Sánchez-García, M. I. Sobrón, and B. Vitoriano. On the set covering polytope : Facets with coefficients in {0, 1, 2, 3}. *Annals of Operations Research*, 81 :343–356, 1998. 68

[54] J. Silberholz and B. Golden. The generalized traveling salesman problem : a new genetic algorithm approach. *Extending the Horizons : Advances in Computing, Optimization and Decision Technologies*, 37 :165–181, 2007. 4

[55] J-C. Simms. Fixed and mobile facilities in dairy practice. *Veterinary clinics of North America - Food animal practice*, 5 :591–601, 1989. 8

[56] P. Slavík. On the approximation of the generalized traveling salesman problem. Technical report, Departement of Computer Science, SUNY-Buffalo, 1997. 4

[57] L-V. Snyder and M-S. Daskin. A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research*, 174 :38–53, 2006. 4, 5

[58] S-S. Srivastava, S. Kumar, R-C. Garg, and P. Sen. Generalized traveling salesman problem through n sets of nodes. *CORS Journal*, 7 :97–101, 1969. 4

[59] W. Swaddiwudhipong, C. Chaovakiratipong, P. Nguntra, P. Lerdlukanavonge, and S. Koonchote. Effect of a mobile unit on changes in knowledge and use of cervical cancel screening among rural thai women. *International journal of epidemiology*, 24 :493–498, 1995. 7

[60] M-F. Tasgetiren, P-N. Suganthan, and Q-K. Pan. An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem. *Applied Mathematics and Computation*, 215 :1417–1422, 2010. 5

l'université nantes angers le mans
PÔLE DE RECHERCHE ET D'ENSEIGNEMENT SUPÉRIEUR

ECOLE DES MINES DE NANTES

Thèse de Doctorat

Minh Hoang HA

Modélisation et résolution de problèmes généralisés de tournées de véhicules
Modeling and solving generalized routing problems

Résumé

Le problème de tournées de véhicules est un des problèmes d'optimisation combinatoire les plus connus et les plus difficiles. Il s'agit de déterminer les tournées optimales pour une flotte de véhicules afin de servir un ensemble donné de clients. Dans les problèmes classiques de transport, chaque client est normalement servi à partir d'un seul nœud (ou arc). Pour cela, on définit toujours un ensemble donné de nœuds (ou arcs) obligatoires à visiter ou traverser, et on recherche la solution à partir de cet ensemble de nœuds (ou arcs). Mais dans plusieurs applications réelles où un client peut être servi à partir de plus d'un nœuds (ou arc), les problèmes généralisés qui en résultent sont plus complexes.
Le but principal de cette thèse est d'étudier trois problèmes généralisés de tournées de véhicules. Le premier problème de la tournée sur arcs suffisamment proche (CEARP), comporte une application réelle intéressante en routage pour le relevé des compteurs à distance ; les deux autres problèmes, problème de tournées couvrantes multi-véhicules (mCTP) et problème généralisé de tournées sur nœuds (GVRP), permettent de modéliser des problèmes de conception des réseaux de transport à deux niveaux. Pour résoudre ces problèmes, nous proposons une approche exacte ainsi que des métaheuristiques. Pour développer la méthode exacte, nous formulons chaque problème comme un programme mathématique, puis nous construisons des algorithmes de type branchement et coupes. Les métaheuristiques sont basées sur le ELS (ou Evolutionary Local Search) et sur le GRASP (ou Greedy Randomized Adaptive Search Procedure). De nombreuses expérimentations montrent la performance de nos méthodes.

Mots clés

Problème de la Tournée sur Arcs Suffisamment Proche, Problème de Tournées Couvrantes Multi-Véhicules, Problème Généralisé de Tournées sur Nœuds, Branchement et Coupes.

Abstract

The Routing Problem is one of the most popular and challenging combinatorial optimization problems. It involves finding the optimal set of routes for fleet of vehicles in order to serve a given set of customers. In the classic transportation problems, each customer is normally served by only one node (or arc). Therefore, there is always a given set of required nodes (or arcs) that have to be visited or traversed, and we just need to find the solution from this set of nodes (or arcs). But in many real applications where a customer can be served by from more than one node (or arc), the generalized resulting problems are more complex.
The primary goal of this thesis is to study three generalized routing problems. The first one, the Close-Enough Arc Routing Problem (CEARP), has an interesting real-life application to routing for meter reading while the others two, the multi-vehicle Covering Tour Problem (mCTP) and the Generalized Vehicle Routing Problem (GVRP), can model problems concerned with the design of bilevel transportation networks. The problems are solved by exact methods as well as metaheuristics. To develop exact methods, we formulate each problem as a mathematical program, and then develop branch-and-cut algorithms. The metaheuristics are based on the evolutionary local search (ELS) method et on the greedy randomized adaptive search procedure (GRASP) method. The extensive computational experiments show the performance of our methods.

Key Words

Close-Enough Arc Routing Problem, Multi-vehicle Covering Tour Problem, Generalized Vehicle Routing Problem, branch-and-cut algorithm, metaheuristic.