Titre: Title:	Optimisation spatio-temporelle des routes pour les problèmes de livraison de colis par services postaux
Auteur: Author:	Alexis Bretin
Date:	2021
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Bretin, A. (2021). Optimisation spatio-temporelle des routes pour les problèmes de livraison de colis par services postaux [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/9108/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Optimisation spatio-temporelle des routes pour les problèmes de livraison de colis par services postaux

ALEXIS BRETIN
Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques

Juin 2021

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Optimisation spatio-temporelle des routes pour les problèmes de livraison de colis par services postaux

présentée par **Alexis BRETIN**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Alain HERTZ, président
Louis-Martin ROUSSEAU, membre et directeur de recherche
Guy DESAULNIERS, membre et codirecteur de recherche
Antoine LEGRAIN, membre
Nicolas ZUFFEREY, membre externe

DÉDICACE

À mes parents, mes premiers modèles, ceux qui ont toujours eu confiance en moi pour m'aider à m'affirmer, ceux qui ont fait la personne que je suis et qui me démontrent tant d'amour malgré la distance. Ceux-là même qui m'ont soutenu, à chaque instant et sans faille, pour rester sur le bon chemin.

À la vie, pour tout ce qu'elle a de beau à nous offrir, ses rires, ses joies et ses moments d'émerveillements.

À la nature, si belle et harmonieuse, mais aussi si fragile, source d'inspiration sans limite que nous devons protéger et préserver aujourd'hui bien mieux qu'hier.

REMERCIEMENTS

Je souhaite tout d'abord remercier mes directeurs de recherche, Guy Desaulniers et Louis-Martin Rousseau, pour l'aide qu'ils m'ont apportée tout au long de ce doctorat, par leur soutien et leur bienveillance. Merci pour votre suivi, vos nombreux commentaires toujours pertinents et vos innombrables relectures (particulièrement Guy) qui ont dû, par période, monopoliser votre temps. Merci de me l'avoir accordé pendant toutes ces années et me permettre, enfin, de conclure ce chapitre enrichissant d'éducation académique.

Merci à la compagnie Giro, et en particulier Charles Fleurent et son équipe avec laquelle j'ai eu la chance de travailler et d'échanger, de m'avoir permis de rendre plus concret mon travail dans le milieu industriel. Merci à eux, ainsi qu'au CRSNG pour leur soutien financier tout au long de mes études doctorales.

J'aimerais aussi remercier Nicolas Zufferey, ainsi qu'Alain Hertz et Antoine Legrain d'avoir accepté d'évaluer mes travaux et d'être sur mon jury de thèse. Je remercie aussi Andrea Lodi qui a présidé le jury de mon oral de présentation de thèse et qui m'a donc permis de continuer mes recherches.

Merci à François Lessard pour son aide capitale dans l'implémentation de nouvelles fonctionnalités dans GENCOL, sa disponibilité et ses retours positifs. Cela a été un plaisir de travailler avec toi.

Merci à Adil Tahir qui m'a épaulé au cours du dernier chapitre et avec qui nous avons eu de nombreuses séances d'idéation très enrichissantes.

Merci au personnel du CIRRELT et du GERAD, en particulier Serge Bisaillon pour son partage de connaissances en termes d'implémentation et Guillaume Michaud pour l'assistance technique et matérielle.

Merci à l'équipe du tarot du GERAD, où on retrouve Guy, mais aussi les fidèles Serge, Alain, François, Matthieu, Sébastien, Carole, Frédéric, Hugues et les autres. Votre accessibilité m'a permis de passer des dîners très conviviaux tout en diminuant significativement mon aversion au risque : nous sommes rarement à bien plus qu'une belle garde-contre de la tête !

Merci bien entendu à Coralie, celle avec qui je partage ma vie depuis bientôt quatre ans déjà, pour son soutien indéfectible dans les hauts et bas d'un doctorat. La vie est plus facile et motivante quand on se sent aimé, merci de me donner tout ça.

Pour rester dans le thème, merci aussi à mes parents qui, malgré la distance et certaines difficultés à me laisser partir si loin d'eux, m'ont toujours encouragé et poussé dans mes

choix sans jamais les remettre en question. Merci à ma sœur, mon frère et aussi mes deux nièces pour les chaleureuses retrouvailles familiales à chacun de mes retours en France.

Merci à mes colocataires, présents et passés, pour leur joie de vivre et les instants de légèreté que nous avons pu passer ensemble, avec une mention spéciale à Rémi, Anaïs et Valentin qui sont devenus bien plus que de simples colocataires.

Merci à mes coéquipiers de basket, Sébastien, Charlotte, Dimitri, Juanma (European Dream Team), Louis, Ian, Alex, Johnny, et tant d'autres passionnés de la balle orange avec qui j'ai pu me vider la tête et épuiser mon corps, mais aussi à coach Richard qui m'a donné tant de confiance en moi.

Merci à Tim', pour tous ces bons moments, notamment quelques longues séances de jeu nocturnes, entremêlées de fous rires et qui m'ont parfois peut-être couté un peu de productivité matinale...

Merci à Florian, qui a traversé le doctorat avec moi et qui se retrouve dans pas mal des groupes sus-cités, qui m'a beaucoup aidé et inspiré par sa méthodologie mais aussi son ouverture au monde.

RÉSUMÉ

En raison d'une forte croissance du commerce en ligne, combinée à un recul significatif du courrier transactionnel, les services postaux ont dû, au cours des dix dernières années, s'adapter à un marché en métamorphose. En effet, le courrier traditionnel et la livraison de colis ne se gèrent pas de la même façon. Pour le courrier postal, les facteurs sont généralement assignés à des zones géographiques dans lesquelles ils doivent visiter toutes les adresses sur toutes les rues au cours de leur tournée. En ce qui concerne les colis, seulement quelques résidents ou commerçants doivent être visités, et l'encombrement des divers colis oblige les livraisons à s'effectuer par camion. Parmi les clients à visiter, certains présentent des contraintes horaires, appelées fenêtres de temps, période durant laquelle leur colis doit être livré. Cette contrainte provient généralement d'accords entre les services postaux et leurs clients (généralement des entreprises). Depuis quelques années, des contrats de livraison en moins de 24H ou 48H peuvent aussi être à l'origine de ces fenêtres de temps.

Le problème de livraison de colis dans le milieu postal possède quelques caractéristiques qui lui sont propres. Le nombre de clients par route est relativement conséquent (traditionnellement entre 50 et 150 clients par route) et la capacité des camions n'est généralement pas limitante. En revanche, la durée des tournées est une mesure essentielle dans ce genre de problème, la durée des journées de travail des livreurs en dépendent. Il est bon de noter que le temps de service, comprenant entre autres stationnement du véhicule, récupération du colis dans le camion et livraison en mains propres, compte pour une part conséquente de la durée des routes et que ce temps est incompressible.

Au cours des dernières années, les compagnies postales ont cherché à assigner à leurs livreurs des routes plus compactes pour diverses raisons. Cela permet notamment d'augmenter la connaissance du quartier des livreurs (travaux, embouteillages réguliers aux heures de pointe, etc.) et ainsi améliorer leurs performances. Dans certaines compagnies pour lesquelles les livreurs reçoivent une récompense pour chaque colis livré, cela peut aussi permettre un retour plus rapide chez un client absent.

Dans cette thèse, nous présentons les outils mis en place pour optimiser les caractéristiques spatio-temporelles des routes dans ce contexte postal. Ce dernier étant très dynamique, puisque les clients peuvent changer d'un jour à l'autre, nous cherchons à développer des méthodes donnant les meilleurs résultats possibles en peu de temps, et ce malgré la taille conséquente des instances. L'intégralité des projets présentés par la suite a été réalisée en collaboration avec Giro Inc., la compagnie montréalaise ayant développé GeoRoute, un logiciel

d'optimisation utilisé par de nombreuses compagnies postales à travers le monde.

Dans la première partie de cette thèse, nous expliquons en quoi la durée des routes peut être un avantage aussi bien stratégique que purement économique. En effet, en nous intéressant au problème de voyageur de commerce avec fenêtres de temps, nous mettons en place des méthodes pour essayer de trouver le meilleur équilibre économique entre temps de parcours et durée de la tournée, le temps des livreurs étant une ressource précieuse et limitée. En plus de ceci, nous développons une méthode permettant de résoudre des instances postales de très grande taille (jusqu'à près de 500 clients) avec très peu de fenêtres de temps en mélangeant agrégations de clients, programmation par contraintes et programmation en nombres entiers. Nous observons que, dans un contexte où il existe des fenêtres de temps, le chemin le plus court n'est pas forcément le moins coûteux, car il faut prendre en considération les éventuels temps d'attente.

Dans le deuxième projet, nous nous intéressons particulièrement à la notion de compacité. À partir d'une compacité géographique considérée comme exacte et calculée comme étant l'aire de l'enveloppe convexe des clients d'une route, nous proposons deux approximations de cette mesure. Par la suite, nous les testons, aussi bien sur des instances académiques que sur des instances dérivées de données du monde industriel, en ajustant un algorithme combinant recherche à voisinage large et génération de colonnes heuristique. Dans ce projet, nous nous limitons à des instances de taille moyenne, relativement à la nature du problème, c'est-à-dire ne dépassant pas les 500 clients. Nous pouvons analyser à travers les résultats obtenus sur les instances académiques que, lorsqu'il existe de nombreuses fenêtres de temps, la compacité des routes ne peut être améliorée qu'au détriment d'un temps de parcours plus long. En revanche, quand les fenêtres de temps sont moins nombreuses comme c'est le cas avec les données réelles, la prise en compte de la compacité peut avoir l'effet d'un guide pour notre méthode, permettant ainsi, en un temps limité, d'améliorer à la fois la compacité et le temps de parcours de chacune des routes.

La troisième partie de cette thèse est une extension du projet précédent, appliqué à des instances industrielles de très grande taille (jusqu'à plus de 3000 clients). Gérer de telles instances amène de nouveaux défis. Nous présentons donc plusieurs techniques pour rendre les temps de calcul raisonnables sans impacter la qualité des solutions. Tout d'abord, nous limitons l'espace de recherche de façon dynamique autour de la solution courante afin de favoriser l'obtention de routes améliorantes. Ensuite, nous utilisons une méthode de décompositions successives du problème, à taille variable pour limiter les effets de bord, pour intensifier la recherche dans toutes les zones géographiques de l'instance. Nous montrons alors que notre heuristique permet d'obtenir des routes plus compactes sans dégrader la qualité de la solution

par rapport à d'autres méthodes négligeant l'aspect compacité.

ABSTRACT

In the last ten years, postal services had to adapt to a changing market, due to vigorous growth in online shopping, combined with a significant decline in transaction mailing. Indeed, traditional mail and parcel delivery are not handled similarly. For the first one, postmen are generally assigned to geographical areas and must visit every address on every street within their assigned territories. For the latter one, only a few residents or businesses have to be visited. Also, truck usage is mandatory because of the bulkiness of the parcels. Time constraints, called time windows are assigned to some of the clients, restricting the visiting time to a predefined period. Those constraints most of the time originate from agreements between the postal service and its clients, generally companies. In recent years, contracts such as 24H or 48H-delivery may also lead to time windows.

The parcel delivery problem has some special characteristics in the postal context. The number of customers per tour is relatively high (50 to 150 customers per route) and the truck capacity is not considered. However, route duration is a key component in this problem, as the postmen shifts depend on it. Furthermore, the service time counts for a substantial part of the route duration. It includes parking time and parcel handling at the customers among other little tasks and is, by definition, incompressible.

In the last years, postal companies also tended to assign more compact routes to their employees for several reasons. It helps to increase the postman's knowledge of the neighborhood (road works, rush-hour congestion, etc.) and thus improve his performance. In some companies where rewards are granted for every parcel effectively delivered, it may enable a faster return to absent clients.

In this thesis, we present tools to optimize spatio-temporal characteristics of the routes in a postal context. The latter is very dynamic, as customers potentially change every day. We intend to develop methods able to generate good quality solutions in a reasonable amount of time, despite the large size of instances. Giro Inc., the Montreal-based company that develops GeoRoute, a software optimizing routes of several postal services worldwide, has supported us during the three different projects presented in the following document.

In the first part of this thesis, we illustrate how dealing with route duration may be beneficial from a strategic point of view as well as for economical considerations. Working on the traveling salesman problem with time windows, we define solution methods that find the best balance between travel time and route duration, deliverers' time being a precious and limited resource. Besides, we develop a method to solve large postal instances (up to about

500 customers) with very few time windows by mixing customer clustering, constraints programming and mixed-integer programming. Finally, we show that when some time windows are set, the shortest path is not necessarily the most cost-effective, as some potential waiting time may occur.

In the second project, we focus on compactness. We first define the exact compactness of a route as the area of the convex hull of the clients belonging to this route. Then, we propose two different approximations of this measure. The two latter approximations are tested on academic benchmarks and on instances derived from industrial instances by adjusting an algorithm combining large neighborhood search and heuristic column generation. In this project, the size of the tested instances is limited to 500 customers, which is not as large as real world-instances in the postal context. We observe that with academic instances, where many time windows are imposed, compactness may be largely improved, but generating more compact routes is costly as longer travel times are required. However, when there are fewer time windows as in real-world instances, considering compactness may even work as a good guide for our method. Indeed, in a limited amount of time, both compactness and travel time may be reduced for the different routes.

The third part of the thesis may be seen as an extension of the previous one, applied to large industrial instances (up to more than 3000 customers). Dealing with such large instances brings new challenges. Therefore, we present several techniques to obtain good-quality solutions in a reasonable amount of time. First, we limit the search space dynamically around the current solution to help to generate improving routes. Then, we use an iterative problem decomposition, with variable size to limit side effects, in order to intensify the search in every geographical zone of the instance. We show that our heuristic generates more compact routes without degrading travel time, compared to other solutions not considering compactness.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

CHAPITRE 1 INTRODUCTION

Avec l'émergence des boutiques en ligne et l'omniprésence des courriers électroniques dans les communications au cours des dernières décennies, les services postaux doivent faire face à une métamorphose majeure de leur quotidien. Le volume de courrier transactionnel s'est effondré (-32% de 2006 à 2015 au Canada [1]) au profit d'un volume de colis croissant ne semblant pas réellement s'essouffler (+9% pour la seule période 2014 à 2015 au Canada [1]). En 2019 et pour la première fois de son histoire, les produits issus de la livraison de colis chez Postes Canada ont été plus élevés que ceux du courrier transactionnel [2]. La même année, la France voyait son volume de colis délivrés augmenter d'encore 6.6% par rapport à l'année précédente [3]. Si l'année 2020 risque d'être bien au-delà des prédictions pour le secteur des colis en raison de la pandémie mondiale Covid-19 (des augmentations régulièrement estimées de 30% à 50% en France [4] ou au Canada [5]), les tendances observées jusqu'à maintenant semblent montrer que la livraison de colis restera le cœur d'activités des compagnies postales pour encore de nombreuses années.

Le problème de livraison de colis dans le milieu postal possède de nombreuses caractéristiques qui le distinguent à la fois de la livraison de courrier transactionnel, mais aussi de problèmes de tournées de véhicules dans d'autres secteurs.

Tout d'abord, nous avons les contraintes liées au véhicule. Contrairement au secteur du courrier où certaines zones urbaines peuvent être desservies par des facteurs à pied ou à bicyclette, l'utilisation d'un camion est inévitable pour la livraison de colis. Cependant, bien que le nombre de clients visités par route soit conséquent (souvent une centaine de clients), le volume intrinsèque de chaque colis étant relativement limité en moyenne, la capacité du camion n'est en pratique pas une contrainte pour la création des routes dans ce contexte.

Il y a ensuite des spécificités relatives aux clients. En effet, la présence de ceux-ci est souvent nécessaire pour compléter la livraison. Afin de faciliter la synchronisation entre les livreurs et les clients, il n'est pas rare de créer des fenêtres de temps chez certains d'entre eux, généralement des entreprises. Il est possible que certaines fenêtres de temps soient aussi créées pour respecter des délais de livraison, typiquement des contrats de livraison en moins de 24H ou 48H. Cependant, afin de garder une certaine flexibilité dans les tournées, mis à part les deux cas de figure cités précédemment, il n'est pas pertinent d'avoir des fenêtres de temps pour tous les clients. Cela a pour impact de complexifier le problème. En effet, la ressource de temps doit être prise en considération et la combinatoire de ce problème est très grande. Avec peu de clients ayant une fenêtre de temps (rarement plus de 25%), aucun

prétraitement n'est réellement très efficace.

La capacité des véhicules n'étant pas contraignante combinée au fait que de nombreux clients n'aient pas de fenêtre de temps pourrait permettre la génération de routes très longues. Cependant, il existe dans le domaine de la livraison de colis en milieu postal une ressource cruciale : le temps de travail du livreur [1]. Celui-ci est limité, de par la nature de son contrat et/ou la législation en vigueur et doit être contrôlé lors de la création des routes, si ce n'est minimisé. En effet, comme nous le verrons dans le chapitre 4, chercher à minimiser la distance parcourue dans un contexte avec des fenêtres de temps n'est pas forcément optimal d'un point de vue économique. À l'époque de la rédaction de cet article, en 2018, nous avions estimé qu'une heure de déplacement urbain d'un véhicule postal coûtait approximativement 8$ (amortissement du véhicule inclus) alors que le salaire horaire des livreurs, chez Postes Canada par exemple, était d'environ 20$ (aujourd'hui, il est plus proche de 22$/h (voir Indeed [2])). Nous pouvons rapidement comprendre que réduire la durée des tournées, en réduisant d'éventuels temps d'attente, permet d'obtenir des routes plus économiques, même si cela doit se faire au prix d'une route un peu plus longue, ce qui n'est *a priori* pas intuitif. Il est bon de noter que le temps de service (stationnement du véhicule, récupération du colis dans le camion, livraison en main propre, ...) compte d'ailleurs pour une part conséquente dans la durée des tournées, étant donné leurs nombres d'occurrences, et que ce temps est incompressible.

Tout au long de cette thèse, notre partenaire industriel Giro, une compagnie montréalaise qui développe des solutions logicielles notamment pour les services postaux, nous a présenté certains de ses nouveaux défis au cours des dernières années. Parmi eux, la notion de routes compactes. S'il est relativement naturel de se faire une idée de ce que serait une route compacte, il ne semblerait cependant pas y avoir dans la littérature de définition universelle qui fasse l'unanimité. Parmi les définitions possibles et en accord avec Giro, nous avons conclu que la définition la plus générale de la compacité d'une route revenait à mesurer l'aire de l'enveloppe convexe de tous les clients d'une route (le dépôt n'étant pas un client, il n'est pas considéré dans ces mesures de compacité). Cette définition a l'avantage de permettre une très bonne représentation visuelle de routes compactes. Dans la pratique, celles-ci peuvent avoir de nombreux avantages. Tout d'abord, elles peuvent permettre une meilleure connaissance du quartier de la part du livreur qui y opère si ce dernier y est régulièrement affecté. Si un client est absent, travailler avec des routes compactes permet aussi un retour plus efficace chez ce dernier ou au comptoir de remise (*e.g.* un bureau de poste ou un commerce de quartier

1. Notons que dans la suite du document, l'emploi masculin sera de mise pour généraliser le personnel affecté à la livraison de colis.

2. https://emplois.ca.indeed.com/cmp/Canada-Post/salaries

partenaire) quand la politique de la compagnie postale le permet. Les routes compactes ont aussi des vertus psychologiques. Bien que difficiles à quantifier, elles procurent un sentiment d'efficacité aussi bien pour les livreurs que pour les planificateurs. La recherche de routes compactes a en plus pour effet secondaire de naturellement limiter le chevauchement des routes, source de frustration chez certains livreurs quand un de leur collègue livre un colis dans une zone qu'ils ont aussi couverte.

Nous pouvons voir que la compacité est ici définie de manière purement géographique, à partir des coordonnées des points et donc naturellement de distances euclidiennes. Mais les véhicules se déplacent eux sur un réseau routier existant et les distances parcourues entre deux clients peuvent parfois différer grandement des distances à vol d'oiseau. Nous pensons notamment à des cas particuliers comme celui où deux clients sont situés de part et d'autre d'un cours d'eau, mais sans présence de pont proche, ou aux contraintes de circulation de certaines zones urbaines (interdiction de tourner à gauche, voie à sens unique, ...).

L'objectif de cette thèse est donc de développer des algorithmes permettant de gérer au mieux les caractéristiques spatio-temporelles des routes dans un contexte de livraison de colis en milieu postal. La composante temporelle est relative à la durée des routes qui est une contrainte dure quant à la réalisabilité de nos solutions. La composante spatiale, elle, est attribuable à la compacité des routes. Contrairement à la durée des routes qui peut avoir un intérêt économique, la compacité n'apporte pas réellement de gains quantifiables perceptibles pour les gestionnaires. C'est avant tout une affaire de préférence et il nous faudra savoir trouver un équilibre entre cette mesure de compacité et le coût réel des routes pour que son intérêt persiste. L'environnement dans lequel nous avons travaillé étant aussi très dynamique (les clients à livrer d'un jour à un autre ne sont définitivement pas tous les mêmes), nous avons apporté un intérêt particulier à développer des méthodes qui ne consomment pas trop de temps de calcul, et ce malgré la très grande taille des instances.

Cette thèse est structurée de la manière suivante. Le chapitre 2 est une revue de littérature autour du problème de livraison de colis dans le milieu postal. L'organisation détaillée de la thèse est présentée dans le chapitre 3. Le chapitre 4 présente l'article "*The traveling salesman problem with time windows in postal services*" publié dans *Journal of the Operational Research Society* (voir [6]). Le chapitre 5 présente l'article "*Compact routes for parcel delivery by postal services*" qui a été soumis à *Computers and Operations Research*. Le chapitre 6 traite des différentes techniques envisageables pour résoudre ce même problème sur des instances de très grande taille. Dans le chapitre 7, nous synthétisons l'ensemble des travaux de recherche effectués au cours de cette thèse, analysant leurs limites et proposant des améliorations futures possibles. Le chapitre 8 viendra conclure cette thèse.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre balaie la littérature sur les problèmes de livraison de colis dans le milieu postal. Il sera décomposé de la manière suivante. Dans une première partie, nous nous intéresserons au problème de tournées de véhicules, en mettant l'accent sur les caractéristiques du milieu postal. Dans un deuxième temps, nous verrons les méthodes de résolution couramment utilisées pour résoudre ce type de problèmes.

2.1 Caractéristiques du milieu postal

Dans cette section, nous verrons les principales différences qui distinguent la livraison postale des autres types de livraisons courants. Nous verrons d'abord le problème assez général de tournées de véhicules avant de mentionner quelques spécificités propres au milieu postal. Nous détaillerons par la suite quelques aspects spatio-temporels des routes déjà présentés dans la littérature, parfois dans des problèmes connexes. Enfin, dans un but pratique, nous nous intéresserons rapidement aux alternatives à la livraison par camion qui commencent à émerger. Ces nouvelles méthodes restent cependant marginales en termes de volume traité.

2.1.1 Le problème de tournées de véhicules avec fenêtres de temps

Le problème des livraisons de colis dans le milieu postal appartient à la classe des problèmes de tournées de véhicules (VRP pour *Vehicle Routing Problem*) introduite en 1959 [7]. Dans ce problème, chaque client doit être visité une et une seule fois par un véhicule appartenant à une certaine flotte prédéfinie, tout en minimisant la distance parcourue par l'ensemble des véhicules. Dans le cas de notre problème, les visites peuvent être contraintes dans le temps avec des fenêtres de temps, laissant naître le problème dérivé de tournées de véhicules avec fenêtres de temps (VRPTW pour *Vehicle Routing Problem with Time Windows*). Ce problème peut aussi être vu comme une extension du problème de voyageur de commerce avec fenêtres de temps (TSPTW pour *Traveling Salesman Problem with Time Windows* en anglais) avec plusieurs véhicules. De manière générale, la classe des VRP a été très largement étudiée et de nouvelles contraintes pratiques ne cessent de s'ajouter au problème initial. Les plus courantes sont détaillées dans [8]. Nous pouvons notamment citer les contraintes de capacité, la nature des flottes (homogènes ou hétérogènes), le nombre de dépôts ou encore le concept de ramassage et livraison (*Pickup/Delivery*). En plus de ces nombreuses variantes pratiques, les problèmes traités peuvent aussi varier de bien d'autres façons.

La nature des fonctions objectif différencie parfois des problèmes *a priori* très similaires. Selon une analyse taxonomique réalisée en 2016 [9] sur 277 articles de VRP publiés entre 2009 et 2015, la minimisation de la distance parcourue (ou du temps de parcours) semble un incontournable (92% des articles étudiés). Le nombre de véhicules utilisés est aussi régulièrement pris en compte (dans 38% des cas). Ceci amène généralement à résoudre des problèmes multiobjectifs [10]. Les objectifs étant parfois contradictoires, la recherche des meilleures solutions passe par l'obtention de fronts de Pareto [11]. Une approche des problématiques sociales ou environnementales amène aussi de nouveaux objectifs. L'équilibrage des distances parcourues pour chaque route [12] ou la minimisation de la pollution engendrée par ces déplacements [13] font partie des alternatives déjà envisagées. D'un point de vue temporel, la présence de fenêtres de temps peut aussi inciter à minimiser les temps d'attente. Plus de détails concernant la durée des routes seront présentés en section 2.1.2.1.

En plus de toutes les problématiques déterministes énoncées jusque là, il existe de nombreuses variantes dynamiques à ce problème [14]. La présence des clients peut être incertaine [15] ou la quantité à livrer inconnue [16]. Le recours, en cas d'absence d'un client ou de demande nulle, est souvent d'ignorer ce client dans une route prédéfinie. Ce problème peut très bien être utilisé pour créer des territoires stratégiques, adaptés quotidiennement pour répondre à la demande du jour. D'autres données peuvent être la source d'incertitudes majeures, notamment le temps de parcours [17].

Le problème de livraison de colis dans le milieu postal s'inscrit dans un cadre de VRPTW aux caractéristiques particulières. La capacité des véhicules n'est pas limitante (contrairement à plus de 90% des papiers sur le VRP [9]). Ceci ne rend pas le problème nécessairement plus simple. La combinaison demandes/capacités permet souvent de calculer une borne inférieure pertinente sur le nombre de véhicules à utiliser, en plus de permettre la génération d'inégalités valides [18]. La proportion de clients possédant des fenêtres de temps, que nous appelons aussi densité de fenêtres de temps, est dans le même temps relativement faible. Une fois encore, le problème devenant moins contraignant pourrait sembler plus facile à résoudre, mais il faut aussi comprendre que moins de prétraitement est possible. Nous pensons notamment aux éliminations d'arcs, qui dans le cadre du TSPTW, peuvent être encore plus poussées [19, 20]. La faible densité de fenêtres de temps augmente donc l'aspect combinatoire du problème.

2.1.2 Les contraintes spatio-temporelles

Au cours de cette thèse, nous avons mis l'accent sur les caractéristiques spatio-temporelles des routes. Nous allons donc présenter dans un premier temps divers articles relatifs à la durée des tournées, avant de nous intéresser à la compacité des routes générées.

2.1.2.1 La durée de tournée

Notre partenaire industriel nous a avertis dès le début de la thèse de l'importance de respecter la durée maximale des routes, relativement au temps de travail autorisé pour les livreurs. S'il est tout à fait possible de chercher à minimiser les durées de routes, il est plus courant de gérer la durée des routes comme des contraintes. Dans un contexte comprenant des fenêtres de temps, le problème diffère légèrement de la variante appelée *Distance-Constrained VRP* [21]. En effet, si pour un déplacement donné, distance et temps sont souvent étroitement liés (ce qui est encore plus vrai dans un contexte déterministe), ce n'est pas le cas lorsqu'il y a des temps d'attente (véhicule à l'arrêt). Pour une séquence donnée, ce temps peut parfois être compressé, en ajustant les temps de départ et de retour au dépôt [22]. Pour les VRPTW classiques, la minimisation de la durée de routes a été relativement peu étudiée. Dans des problèmes à un seul véhicule [23, 24], il semblerait que cette fonction objectif seule soit plus simple à traiter que de minimiser la distance parcourue, mais les modèles utilisés ne permettent pas de différer l'heure de départ du dépôt. Cet objectif a cependant connu un regain d'intérêt avec les problèmes comportant des fenêtres de temps multiples, que ce soient des VRP [25] ou des problèmes voisins [26]. Cette gestion de la durée des routes se complexifie aussi si les temps de parcours le long des arcs dépendent de l'heure de passage, comme dans les TDVRP (pour *Time-Dependent VRP*) [27] servant notamment à modéliser le concept d'heure de pointe. Concernant les contraintes de durée de routes, nous pouvons noter que la méthode de recherche taboue [28] a été améliorée quelques années plus tard [29] justement pour mieux prendre en considération les contraintes de durée de routes.

2.1.2.2 La compacité

La mesure de la compacité des routes est un des pans essentiels de cette thèse. Dans la littérature, qui n'est pas si dense à ce sujet, différentes approches ont déjà été mises à l'essai, que ce soit dans des VRP ou dans des problèmes voisins, comme le problème de collecte d'ordures [30]. Il est ici question d'une mesure basée sur la somme des distances euclidiennes entre les différents points visités et le centre de gravité de la route à laquelle ils appartiennent. De plus, les auteurs ont aussi cherché à minimiser le chevauchement des enveloppes convexes des routes, un principe en adéquation avec l'objectif de définir des routes compactes. Cependant, une mesure basée sur une somme de distances a un inconvénient : visiter deux clients exactement au même endroit (par exemple deux colis distincts dans un même immeuble) affectera deux fois la compacité de la route. On retrouvera une idée similaire appliquée au problème de découpage de territoires de ventes [31], avec comme point de référence non plus le centre de gravité de la route, mais un client lui appartenant, limitant donc ces localisations possibles,

tout en ajoutant des variables de décision au problème.

Certains se sont aussi demandé si la compacité pouvait être un gage de qualité pour trouver de bonnes routes [32]. Ils ont alors défini deux autres mesures de compacité basées sur une ligne fictive de référence passant par le dépôt et le centre de gravité de la route. La première correspond à la somme des distances euclidiennes entre chaque client visité et cette ligne, la deuxième à la somme des angles entre, les demi-droites passant par le dépôt et les clients d'une part et la ligne de référence d'autre part. En plus d'être des mesures additives, elles ne semblent pas les plus appropriées au contexte postal, car elles auront plutôt tendance à former des routes en formes de "pétales" qui ne sont pas vraiment recherchées dans nos travaux. Cependant, leurs travaux ont montré que notre intuition était bonne dans un contexte sans fenêtre de temps. Les solutions proches de l'optimalité sont souvent parmi les plus compactes.

La compacité géographique suscite aussi de l'intérêt dans des domaines un peu plus lointains, comme le découpage électoral, pour lequel de nombreuses mesures ont été comparées [33]. Certaines d'entre elles ne dépendant que de la forme des districts eux-mêmes. Nous y retrouvons des mesures basées sur leur aire et/ou leur périmètre, mais aussi sur diverses formes encapsulantes (rectangle, hexagone, cercle) ou encore quelques mesures plus complexes comme des calculs de moment d'inertie sur les formes des districts. Parmi les 34 mesures de compacité mises au banc d'essais dans cet article, aucune ne fait l'unanimité, car il semblerait impossible de pouvoir traiter correctement toutes les exceptions possibles. Les auteurs rappellent que la compacité d'une forme est bien souvent basée sur un ressenti visuel, propre à chaque individu.

2.1.3 Variantes technologiques pour la livraison de colis postaux

Dans les dernières années, de nombreuses alternatives à la livraison par camions sont apparues. Tout d'abord, des points de services ont été ajoutés. Il est dorénavant parfois possible de se faire livrer dans un commerce de proximité ou dans un casier à colis [34]. Dans certaines approches, les clients peuvent même définir des préférences de lieu de livraison en fonction de l'heure de la journée [35]. Le développement des drones a aussi contribué à l'apparition de nouvelles méthodes de livraison, ceux-ci étant généralement combinés à un camion servant de base de lancement [36, 37]. Le camion reste utile pour couvrir un bon nombre de clients et est indispensable de par sa capacité, comparativement à celle d'un drone. Ces derniers peuvent cependant livrer quelques clients isolés, permettant d'éviter au camion des détours peu efficaces. Si sur le papier, cette méthode de livraison semble très pertinente, de nombreuses contraintes, notamment légales, par rapport à l'utilisation de drones, empêchent son application plus globale pour le moment.

2.2 Méthodologies

Dans cette section, nous aborderons différentes méthodologies employées pour résoudre des problèmes proches de celui que nous traitons. Nous présenterons tout d'abord certaines méthodes exactes. Nous nous attarderons ensuite sur les méthodes heuristiques qui sont bien plus appropriées à la taille des problèmes que nous résolvons. Enfin, nous ferons un bref tour de la littérature relative à l'agrégation de données, dans notre cas appliquée aux clients.

2.2.1 Méthodes exactes

Pour résoudre des problèmes aussi complexes que le VRP et ses variantes, les méthodes exactes les plus couramment utilisées reposent sur le principe de séparation et évaluation (BB pour *Branch-and Bound* en anglais). Afin d'améliorer les performances de cette méthode, le BB est régulièrement combiné à l'utilisation de plans sécants (BC pour *Branch-and-Cut*), de la génération de colonnes (*BP pour Branch-and-Price*) ou de la combinaison des deux (BPC pour *Branch-and-Price-and-Cut*). La plupart des modélisations de BPC sont présentées dans l'enquête très complète de Costa *et al.* [38]. La génération de colonnes comprend un problème maitre et un sous-problème, sous forme de problème de plus court chemin élémentaire avec contraintes de ressources. Ce dernier, NP-difficile, doit être résolu de manière exacte pour garantir l'optimalité des solutions finales trouvées. La méthode utilisée repose essentiellement sur de la programmation dynamique, de sa forme initiale [39], bidirectionnelle [40] ou d'autres variations [41]. Ce sous-problème demeurant très complexe, la contrainte d'élémentarité est parfois relaxée, totalement ou partiellement. L'élémentarité peut n'être imposée qu'à un sous-ensemble de clients [42] ou par l'intermédiaire de voisinages comme avec les *ng-paths* [43]. Cependant, pour accélérer la recherche de nouvelles colonnes, des heuristiques sont parfois utilisées, que ce soit encore par de la programmation dynamique (heuristique cette fois), des méthodes taboues [42] ou autres.

2.2.2 Méthodes heuristiques

Pour résoudre les VRPTW, il existe deux grandes familles de métaheuristiques.

La première est basée sur des voisinages. Tout d'abord, nous y retrouvons les méthodes taboues, introduites par Glover [44, 45] et applicables à de nombreux problèmes de génération de routes, que ce soient des VRPTW standards [28], avec fenêtres de temps flexibles [46, 47] ou encore livraisons fractionnées [48]. Dans ces méthodes, les voisinages sont définis par des opérateurs appelés mouvements. Pour une solution x donnée et un opérateur donné, le voisinage $\mathcal{N}(x)$ est alors défini par l'ensemble des solutions qu'il est possible d'obtenir en appliquant

l'opérateur à x. Parmi elles, la meilleure solution x' est retenue et le processus est réitéré, tout en interdisant le mouvement inverse nous ramenant directement à x. D'autres types de voisinages existent et s'inscrivent dans des cadres algorithmiques distincts. La recherche par voisinage large (LNS pour *Large Neighborhood Search*) [49] consiste, à partir d'une solution courante x, à en détruire une certaine partie avant d'entamer un processus de reconstruction. L'opérateur de destruction comprend généralement un comportement aléatoire et la méthode de reconstruction peut être exacte ou heuristique. Cette dernière peut par exemple prendre la forme d'une simple méthode de meilleure insertion gloutonne mais est parfois plus complexe, pouvant notamment faire appel à de la génération de colonnes [50]. Il y a parfois plusieurs opérateurs de destruction et/ou plusieurs méthodes de reconstruction possibles, qui peuvent être choisis au début de chaque itération en fonction de certaines probabilités. Celles-ci dépendent de leur succès au cours des précédentes itérations pour la résolution en cours. Dans ce cas là, nous parlons alors d'ALNS (*Adaptive Large Neighborhood Search*)[51]. Enfin, il y a les méthodes de recherche à voisinage variable (VNS pour *Variable Neighborhood Search*) [52]. Tout d'abord, une famille de voisinages est définie, par exemple avec une profondeur variable (nous parlons alors de *Variable-depth Neighborhood Search*). Ensuite, à partir, d'une solution courante, une nouvelle solution est générée dans son premier voisinage avant d'appliquer une méthode de recherche locale afin de trouver un optimum local. Si la solution obtenue est meilleure que la solution courante, elle la remplace et le processus recommence. Sinon, le voisinage suivant, plus large et donc plus permissif, est utilisé. Cette méthode est notamment utilisée dans [53], pour une application au VRPTW avec plusieurs dépôts.

La seconde grande famille de méthodes heuristique s'articule autour de populations. Parmi elles, nous retrouvons des approches basées sur les colonies de fourmis [54], mais surtout des algorithmes évolutionnaires [55], notamment génétiques [56]. Pour résoudre des instances de taille intéressante pour un problème aussi complexe, de nombreuses hybridations ont fait leur apparition et montré des résultats très concluants. Nous pensons notamment à l'algorithme de Vidal *et al.* [56] qui combine un algorithme génétique, une recherche locale et des mécanismes de diversification ou celui de Prescott-Gagnon *et al.* [50] qui allie une recherche à voisinage large et une génération de colonnes heuristique. Cette dernière approche, tout comme celle décrite dans [57], se décompose d'ailleurs en deux phases distinctes. La première a pour but de minimiser le nombre de véhicules utilisés, la seconde de minimiser la distance parcourue.

2.2.3 Agrégation

Quand la taille des instances vient à augmenter, il semblerait naturel de chercher des techniques pour se ramener à des instances de taille plus raisonnable. L'une d'entre elles est

l'agrégation de clients [58]. L'objectif étant de rassembler des clients similaires afin de les traiter, dans un premier temps, comme une entité unique et ainsi diminuer la taille de l'instance considérée. La méthode la plus courante d'agrégation s'intitule *k-means* [59] et a pour objectif de déterminer k groupes distincts afin de minimiser la somme des carrés des distances de chaque point au centre de l'agrégat auquel il appartient. Parmi les défauts de ces méthodes, il y a l'identification de la valeur de k [60], l'initialisation des centres [61], mais aussi le fait que le résultat donne une partition de clients avec des agrégats disjoints. Pour pallier ce dernier problème, il peut être pertinent de se tourner vers des agrégats flous [62] dans lesquels les clients ne sont pas affectés à chaque agrégat de façon binaire, mais avec un certain degré d'appartenance.

CHAPITRE 3 ORGANISATION DE LA THÈSE

Comme nous avons déjà pu le voir dans les chapitres précédents, cette thèse a été réalisée avec l'aide de la compagnie Giro, qui a pu nous présenter beaucoup des défis existants dans le domaine postal et en particulier dans la livraison de colis. Giro a de nombreux clients dans divers pays autour du globe, pour lesquels les approches de la livraison de colis peuvent parfois différer. Giro porte une attention toute particulière à répondre à des problématiques souvent propres à chacun de ses clients spécifiques pour lui apporter la solution la plus adaptée. Dans cette thèse, nous avons essayé de dégager des problématiques qui ne seraient pas trop spécifiques afin qu'elles puissent être intéressantes pour n'importe quel acteur du domaine. C'est ainsi que nous avons dégagé la compacité des routes comme un critère pertinent d'évaluation de la qualité des routes.

L'heuristique actuellement utilisée par Giro pour attaquer les problèmes de taille industrielle auxquels elle fait face est composée de deux étapes : il faut identifier dans un premier temps des territoires par rapport à divers critères, avant de déterminer un séquencement précis des clients dans chacun d'entre eux. En anglais, nous parlons de *Cluster First, Route Second* (CFRS).

Dans cette optique, le chapitre 4 s'attaque au problème du voyageur de commerce (TSP pour *Traveling Salesman Problem*) avec fenêtres de temps. L'idée est donc de voir tous les territoires générés dans la phase "*Cluster First*" comme des sous-problèmes indépendants. Tous les clients du territoire devant être visités, la notion de compacité géographique n'a donc ici pas lieu d'être. C'est pourquoi nous avons mis l'accent dans ce chapitre sur la durée des routes, notamment pour mettre en avant que l'impact économique des temps d'attente était bien plus important que celui de faire de petits détours pour les limiter. Au cours de ce chapitre, nous nous sommes appuyés à la fois sur une modélisation en nombres entiers et sur de la programmation par contraintes, cette dernière méthode étant particulièrement rapide et efficace pour minimiser les durées de routes. Le résultat de ces recherches a fait l'oeuvre d'un article intitulé *The traveling salesman problem with time windows in postal services* et publié dans le *Journal of the Operational Research Society* en février 2021 [6].

Par la suite, nous avons étendu notre vision du problème à quelque chose de plus global. L'objectif étant de se détacher de cette approche CFRS pour laquelle les résultats finaux sont grandement dépendants de la qualité du découpage territorial initial. Dans le chapitre 5, nous avons donc attaqué le problème dans son entièreté, tout en nous limitant à un nombre raisonnable de clients (jusqu'à 500 clients par instance). La méthode de résolution utilisée ici

combine de la recherche à voisinage large pour détruire la solution courante et de la génération de colonnes afin de reconstruire les routes précédemment partiellement détruites. Les clients étant amenés à changer de route au cours du processus de résolution, nous avons donc ici introduit la notion de compacité, notamment à travers l'aire géographique de l'enveloppe convexe (et de ses approximations) définie par les clients de chaque route. Ces travaux ont permis la rédaction d'un article intitulé *Compact routes for parcel delivery by postal services* et soumis au journal *Computers and Operation Research* en janvier 2021.

Au moment d'étendre notre méthode à des instances de taille industrielle comptant jusqu'à plus de 3000 clients, nous nous sommes rendu compte des limites de celle-ci. En plus d'une baisse de l'efficacité algorithmique naturelle due à la grande taille des instances, nous avons vu apparaitre d'autres phénomènes problématiques qui n'étaient pas encore apparus jusque là, notamment autour de la durée maximale autorisée pour les routes. Concernant l'efficacité algorithmique, nous nous sommes posé la question des voisinages qu'il était pertinent de considérer à chaque étape de notre méthode. Nous sommes rapidement retombés sur une forme pouvant s'apparenter à du CFRS, mais avons fait en sorte de le rendre plus dynamique et évolutif au cours de la résolution. Nous avons alors introduit la notion de degrés d'appartenance, permettant une évaluation plus souple des voisinages que des méthodes traditionnelles d'agrégation. Nous avons alors pu montrer qu'à partir de solutions initiales convenables ou de solutions déjà optimisées (post-traitement), nous pouvions encore améliorer la compacité des routes sans dégrader la qualité de la solution.

CHAPITRE 4 ARTICLE 1 : THE TRAVELING SALESMAN PROBLEM WITH TIME WINDOWS IN POSTAL SERVICES

Cet article a été écrit par A. Bretin, G. Desaulniers et L.-M. Rousseau et publié dans *Journal of the Operational Research Society* en février 2021 [6]. Certaines modifications ont été apportées, en rouge, par rapport à l'article d'origine. Il s'agit de corrections d'erreurs détectées par les membres du jury, ainsi que de légères précisions permettant de lever quelques ambiguïtés.

4.1 Introduction

In postal services, letters and parcels are not handled in the same way. Although each delivery employee is assigned to a predefined territory, the routes are not managed identically. For letters which are increasingly being replaced by email, the employee must still visit both sides of every street in the assigned area. For parcel deliveries, only a few residents and businesses must be visited. We position our work in the context of a parcel-delivery problem with time windows (TWs) which can be seen as a variant of the vehicle routing problem with TWs (VRPTW). The TWs are designed to synchronize deliveries and recipients. For large-scale instances involving 10 to 20 territories and up to almost 500 deliveries in a territory, this problem can be solved via a greedy cluster-first route-second (CFRS) procedure. The clustering phase determines well-designed territories according to the specific characteristics of the instance, and the routing phase designs the routes. The two phases are carried out alternately in the following greedy fashion. First, a cluster containing a number of service points (customers) whose total demand does not exceed the capacity of a vehicle is proposed. Second, a route is determined for this cluster. If it is too long or too short compared to a usual employee working shift, the set of points proposed for this territory is adjusted and a new route is computed. This process repeats until obtaining a satisfactory route for this territory. When this is accomplished, the algorithm moves on to the next territory. The overall objective is to find a good solution in a limited time, generally at most one minute per territory.

In this paper, we focus on the routing phase of this CFRS procedure, which solves a variant of the traveling salesman problem with TWs (TSPTW). The classical TSPTW consists of finding a route that starts and ends at a depot, visits every customer once within a pre-specified time interval called a TW, and minimizes the total routing cost. As discussed below, the postal services variant of the TSPTW differs from the classical one by considering a bi-objective cost function and no TW for a large proportion of the customers.

In most studies on the TSPTW, the routing cost is proportional to the total distance traveled along the route or the closely related travel time (TT), i.e., the time spent driving along the route, but waiting before the opening of a customer TW is usually not penalized. In postal services, focusing on TT may be costly because the optimal solution may have considerable waiting time, and drivers must be paid when they are waiting, i.e., when they arrive at a service point before the TW opens. Therefore, beside minimizing TT, the objective function also considers minimizing the route duration (RD), i.e., the difference between the end time and the start time of the route or, equivalently, the sum of the TT, the service time and the waiting time. Note that RD can be assimilated to the makespan, which is defined in scheduling problems as the span of time necessary to complete all the tasks. However, it differs from the completion time of the final task (return to the depot) because, when there are TWs, starting the route at time 0 may not be optimal.

Two observations support the consideration of RD in the objective function. First, in a CFRS approach to the VRPTW, minimizing the RD may make it possible to visit more customers on a route, thus reducing the number of territories. Routes that are more compact allow a better knowledge of the neighbourhood, enabling the drivers to make better decisions in the event of unexpected congestion or construction. Note that reducing the RD may also be relevant in other applications. For instance, in armoured-truck routing, the less time a vehicle spends on the road, the safer the operation, and in fresh-food delivery, the less time the goods spend in the vehicle, the better their condition on arrival.

Second, the average wage for a Canada Post delivery driver is around \$20/h (see Indeed) [1]. For a vehicle such as a Ford Transit, the fuel consumption is around 12 L/100km (see Ford) [2]. If the average speed is between 30 and 50 km/h the approximate cost of one hour of driving is $40 \times 12/100 \times 1.2 = \$5.76$ where 1.2 is the average price of a liter of gasoline in Canada (see Natural Resources Canada) [3]. This may be increased to \$8 to take into account the depreciation of the vehicle and the fact that the fuel consumption is generally slightly higher than the manufacturer's nominal value. Therefore, it is profitable to extend the TT by 2 minutes to save 1 minute of RD, and the idle time is reduced by 3 minutes. The ratio may vary, but the analysis suggests that one unit of RD is more valuable than one unit of TT. Reducing RD may also reduce idle time, and so improve driver satisfaction. In summary, the TT relates to the vehicle cost whereas the RD to the human cost of the route. In the following, we denote by γ^{TT} and γ^{RD}, the costs per time unit when the vehicle is moving and when the driver is working, respectively.

1. https://emplois.ca.indeed.com/cmp/Canada-Post/salaries
2. http://www.ford.ca/commercial-trucks/transit-connect-cargo-van/2017/features/capability
3. http://www2.nrcan.gc.ca/eneene/sources/pripri/price_map_e.cfm

To further illustrate why minimizing RD can be interesting, consider the example presented in Figure 4.1, where D represents the depot, and C_l, $l \in \{1, \ldots, 6\}$, the customers. The intervals near the customer nodes are their TWs, whereas the numbers on the arcs are the arc TTs. The service times are assumed to be all equal to 0. The blue (dashed) path is the optimal route when only minimizing the TT. Its TT is $z_{TT} = 15$, but its RD is $z_{RD} = 19$ because the driver cannot leave the depot later than time 2 and return to it before time 21. This path induces a minimal waiting of 4 time units at C_2. The red (solid) path is the optimal solution when only minimizing the RD. This route also leaves depot D at time 2 but instead of visiting C_2 right after C_1 and waiting 4 time units, it visits C_3 between C_1 and C_2. This detour generates a longer TT but decreases the RD by 2. Its TT and RD are equal to $z'_{TT} = 17$ and $z'_{RD} = 17$, respectively. Consequently, the red path allows the replacement of 4 time units of waiting by 2 extra time units of traveling. Using weights equal to the above cost estimations ($\gamma^{TT} = 8$ and $\gamma^{RD} = 20$), these two solutions can be compared with respect to the bi-objective cost function $\gamma^{TT} \times TT + \gamma^{RD} \times RD$. The TT-optimal solution has a value of $15 \times 8 + 19 \times 20 = 500$, whereas the RD-optimal solution a value of $17 \times 8 + 17 \times 20 = 476$, which yields a saving of 4.8%.

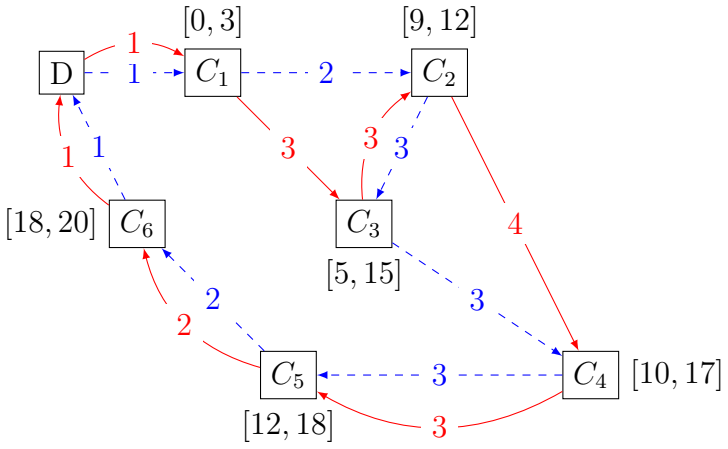


Figure 4.1 Two different solutions: the blue (dashed) route minimizes TT, the red (solid) one minimizes RD

Note, however, that focusing only on RD is not ideal in some cases because extra TT may be incurred when waiting is unavoidable. To see this, let us consider again the example of Figure 4.1 but assuming that the TW at C_6 is $[20, 22]$ instead of $[18, 20]$. In this case, the red path will have to wait 2 time units at C_6, increasing its RD to $z'_{RD} = 19$ and keeping $z'_{TT} = 17$. Given that the statistics for the blue path do not change ($z_{RD} = 19$ and $z_{TT} = 15$), the blue path becomes as good as the red path with respect to RD but remains better with respect to TT and should, therefore, be selected to reduce vehicle costs. This example justifies why

the two criteria need to be considered in the objective function.

TWs are particularly useful for companies, where there are dedicated employees to receive and process deliveries. Furthermore, they often receive express deliveries that have tight delivery deadlines, inducing TWs for the postal companies. On the other hand, because of the rise in online shopping, a growing number of individual customers receive parcel deliveries. Some delivery companies assign TWs to only the largest customers (e.g., stores or industries) and not to individuals. This gives rise to instances in which less than 10% (perhaps just 2% or 3%) of the service points have TWs. This percentage is called TW density and, when it is not high enough, it makes the NP-hard TSPTW even harder to solve, as less preprocessing can be done to reduce the solution space.

The classical TSPTW has been extensively studied. [63] showed that even only finding a feasible solution to it is NP-hard. From branch-and-bound algorithm using a State-space-relaxation approach to compute the lower bounds [64], to dynamically generated time-expanded networks [65], many exact solution algorithms for the TSPTW have been developed. To the best of our knowledge, the most efficient one is that of [66]. They combine column generation to compute lower bounds with dynamic programming to find feasible solutions based on the last bound found, given a certain tolerance.

Some heuristics have also been developed to solve the TSPTW. [67] proposed an insertion heuristic, which is an adaptation of the GENIUS heuristic introduced by [68] for the case without TWs. [69] developed a heuristic that combines the solution of an assignment problem, a greedy insertion procedure and local search. Later, [70] designed a compressed-annealing heuristic, a simulated annealing variant, [71] combined ant colony optimization with beam search, a heuristic that approximates a branch-and-bound method originally developed for scheduling problems, and [72] introduced a variable neighborhood search heuristic.

These papers deal with minimizing TT. [22] was the first to address RD minimization in the TSPTW, by extending the local search heuristic based on edge exchanges that he developed a few years before [73]. Also he did not fix the departure time from the depot, and introduced the concept of forward slack time to help reducing waiting time. According to [23] who minimize the completion time of the route, *i.e.*, with a fixed departure time from the depot, this variant of the problem can generally be solved more quickly than minimizing TT. This is confirmed by the computational results obtained by [24] using mixed-integer programming (MIP) formulations. In these solution approaches, the focus is only on minimizing RD or the completion time, and the TT is let aside.

Note that the benchmark instances used to evaluate these solution methods have generally 100% TW density, enabling efficient preprocessing to reduce the TWs' width, especially

when they are relatively narrow initially. This preprocessing relies mainly on some efficient techniques proposed by [19] and on advanced arc elimination criteria introduced by [20].

To the best of our knowledge, there are no works in the literature that address the TSPTW variant introduced in this paper. Therefore, the contribution of this paper is to devise an effective solution algorithm to tackle large-scale, bi-objective TSPTW instances with low TW density that arise in the postal services. This clustering heuristic exploits the strength of constraint programming (CP) for minimizing the RD and of MIP for minimizing the TT. To enable the solution of large instances within a tight computational budget, it proceeds in steps: it builds a clustered problem, solves it, and disaggregates the clustered solution. To our knowledge, we are the first to handle the combined TT and RD objective function. Therefore, to highlight the possible tradeoffs between RD and TT in the solutions computed for different objective functions (minimizing TT only, RD only, or a linear combination of both), we first report computational results obtained on small and medium-sized TSPTW instances taken from the literature and for which all customers have a TW. Finally, to assess the performance of the proposed clustering heuristic, we present computational results obtained on real industrial instances provided by our research partner.

The rest of this paper is structured as follows. In Section 4.2, we introduce a CP model for the TSPTW where the objective function can be easily adapted to minimize RD or TT or a combination of both. We also summarize a MIP model (fully described in the Appendix 4.A) previously introduced in the literature. In Section 4.3, we describe our clustering heuristic. Computational results on the benchmark instances are reported in Section 4.4 and those on the industrial instances in Section 4.5. Finally, a short conclusion is drawn in Section 4.6.

4.2 Problem formulations

Consider a directed graph $G = (V, A)$ where V contains the nodes, including p and q representing the depot at the beginning and the end of the tour respectively, and A contains the arcs. The TSPTW consists of finding a single tour that visits each node $i \in V$ exactly once while minimizing the routing cost. Some nodes $i \in V$ must be visited within a TW $[R_i, D_i]$, where R_i is the release time and D_i the deadline of node i. For the others, we assume that they must also be visited within an unconstraining TW $[R_i, D_i]$, where $R_i = 0$ and D_i is equal to a very large value. A service time s_i may be associated with each $i \in V$. A TW defines the period in which the service to this node must occur. For node i, $s_i > 0$ indicates that the service could end after D_i but must begin before. Early arrival at node i is allowed, but the driver must wait until R_i. Because of the TWs, set A does not contain all the couples $(i, j) \in V^2$ but only a subset. Indeed, arc (i, j) exists only if $R_i + s_i + t_{ij} \leq D_j$,

where t_{ij} is the traveling time along (i, j). Furthermore, some preprocessing can be applied to reduce the solution space, and thus the combinatorial nature of the problem. TWs may be tightened and node precedence lists built [74]. For each node $i \in V$, the precedence list $V^+(i) = \{j \in V \backslash \{i\} \mid j \prec i\}$ gathers all the nodes j that have to be visited before i. Some arcs may be deleted based on the reduced TWs and the precedence relationships [75].

We first present a CP formulation before explaining the use of bi-objective programming to minimize TT and RD. Then, we summarize a MIP formulation called the time-bucket formulation (TBF).

4.2.1 Constraint programming model

CP is known to be efficient for machine scheduling problems. Since the TSPTW is equivalent to a one-machine scheduling problem with sequence-dependent setup times and TW constraints, a simple but reliable CP formulation is available. We introduce the variable $P_l, \forall l \in \{1, \ldots, |V|\}$, which models the l^{th} service point visited, and T_{P_l}, the time of the visit to this node.

$$\min z = T_q - T_p \tag{4.1}$$

$$\begin{aligned}
\text{subject to:} \quad P_1 &= p && && (4.2)\\
P_{|V|} &= q && && (4.3)\\
P_l &\in V, && \forall l \in \{1, \ldots, |V|\} && (4.4)\\
&\texttt{alldifferent}(P) && && (4.5)\\
T_{P_l} + s_{P_l} + t_{P_l P_{l+1}} &\leq T_{P_{l+1}} && \forall l \in \{1, \ldots, |V| - 1\} && (4.6)\\
T_{P_l} &\in [R_{P_l}, D_{P_l}], && \forall l \in \{1, \ldots, |V|\} && (4.7)\\
T_j &\leq T_i, && \forall i \in V, j \in V^+(i). && (4.8)
\end{aligned}$$

The goal of minimizing the RD corresponds to $\min z = T_{P_{|V|}} - T_{P_1}$, but since we constrain the route to begin and end at the depot via constraints (4.2) and (4.3), we may write the objective function as shown in (4.1). Constraint (4.5) is the well-known global constraint `alldifferent`, extensively detailed in [76], which indicates that the service points visited for $l \in \{1, \ldots, |V|\}$ must be pairwise different (*i.e.*, $P_l \neq P_k, \forall (l, k) \in \{1, \ldots, |V|\}^2$ such that $l \neq k$). Combined with constraints (4.4), this ensures that every service point is visited exactly once. Constraints (4.6) impose sufficient time between two consecutive service point visits (P_l and P_{l+1}) to perform service at P_l and travel from P_l to P_{l+1}. Constraints (4.7) ensure

that the TWs are respected, whereas constraints (4.8) enforce the precedence constraints identified by the preprocessing.

As mentioned in the introduction, we distinguish our notion of RD from the makespan or the completion time, as used in machine scheduling. We define the RD to be the difference between the departure time from and the return time to the depot.

The model can be adapted for the TT problem by replacing (4.1) by

$$\min z = \sum_{l=1}^{|V|-1} t_{P_l P_{l+1}}. \tag{4.9}$$

4.2.2 Multiobjective model: Weighted objective function

For postal services, where the TSPTW is solved many times in a CFRS procedure, we wish to process as many customers as possible within the duration of the driver's shift. However, preliminary results have shown a significant reduction in the quality of the TT when it does not appear in the objective function. It sometimes causes a slight increase in the cost of the final solution. Here we are referring to the cost in dollars, calculated via the parameters γ^{TT} and γ^{RD}. The reduction in the RD does not always compensate for the substantial increase in the TT. This led us to explore how to make the solution more profitable while trying to get the RD as low as possible.

There are many algorithms for multiobjective optimization problems; see [77]. A simple approach is to use a weighted objective function (WOF) that combines the various objectives into a single function. The challenge is to appropriately weight each objective. We decided to use the costs γ^{TT} and γ^{RD} as the weights. The CP model remains the same except that the objective function becomes

$$\min \quad \gamma^{TT}\left(\sum_{l=1}^{|V|-1} t_{P_l P_{l+1}}\right) + \gamma^{RD}(T_q - T_p). \tag{4.10}$$

The main drawbacks of this model are the sensitivity to the costs γ^{TT} and γ^{RD} and the lack of control of the RD objective function. Moreover, the TT part of this WOF makes optimality harder to reach with this CP-model (as illustrated in Table 4.7 of Appendix 4.B).

4.2.3 A time-bucket formulation

The TBF introduced by [75] is a MIP model that partitions the TW associated with each node into time buckets and uses arc flow variables indexed by the time bucket containing the

start traveling time along the arc. These time buckets provide an approximate modeling of the TWs that is a relaxation of these constraints and whose accuracy depends on the width of the buckets. Subtour elimination constraints (SECs) and infeasible path cuts (IPCs) are, thus, required to eliminate solutions that do not meet the TW restrictions and are not filtered out by the time bucket modeling. Compared to a time-indexed model, the main advantage of the TBF is a significant reduction in the number of variables when the buckets are sufficiently large. On the other hand, to get a good relaxation of the TW restrictions and avoid generating many SECs and IPCs, they should not be too large. Details on the TBF can be found in the Appendix 4.A.

[75] showed that the TSPTW can be efficiently solved by a branch-and-cut algorithm applied to the TBF when the objective aims at minimizing TT. As discussed in the Appendix 4.A, it is more difficult to use the TBF to solve the TSPTW when the objective function consists of minimizing RD. Indeed, it might require a complete discretization of the TWs at the depot nodes p and q.

4.3 Solution algorithm for real-world instances

The CP models presented in Sections 4.2.1 and 4.2.2 can be solved using a commercial CP solver. Furthermore, the TBF discussed in Section 4.2.3 can be solved using a commercial-based branch-and-cut algorithm as described in the Appendix 4.A. Preliminary computational tests have shown that the CP algorithm is efficient at finding, in short computational times, high-quality solutions for small and medium-sized instances when minimizing the RD (see Figure 4.6 in Appendix 4.B). On the other hand, it struggles for large instances or for minimizing the TT. At the opposite, the TBF-based algorithm is much better at minimizing the TT than the RD. Its efficiency is, however, highly dependent on the tightness of the TWs. Finally, as for the CP algorithm, the TBF-based algorithm cannot solve large instances in relatively short computational times. Therefore, both algorithms cannot be used successfully to solve large-scale TSPTW instances with a bi-objective function and very few TWs. Computational results supporting these statements are presented in Appendix 4.B.

As discussed in the introduction, priority is given to minimizing the RD over the TT, but minimizing the TT must not be neglected. In a context without TWs, minimizing the RD is equivalent to minimizing the TT because waiting is always avoidable. When there are few TWs, minimizing the RD also helps to reduce the TT: the idle time in the solution is sparse, so every improvement in the RD is generally obtained by reducing the TT. On the other hand, minimizing the TT does not necessarily yield a solution with a low RD because waiting is not penalized at all. This led us to develop, for large-scale postal services'

TSPTW instances, a solution approach that does not consider directly a weighted sum of the two objective functions, but rather combines CP, which is really efficient to minimize the RD, with MIP, more likely to close the optimality gap while minimizing TT.

In this section, we present the proposed clustering heuristic for solving large-scale, real-world, postal services' TSPTW instances. This heuristic proceeds in three steps. First, it clusters some service points, among the ones without a TW, to reduce the size of the instance, yielding a so-called clustered problem that approximates the original one. Second, it solves this clustered problem. Finally, given the sequence of the service points in the computed tour, the clusters are disaggregated to derive a solution to the original instance. These steps are described in the following subsections.

4.3.1 Clustering

Traditional models use the TWs to reduce the solution space; we instead cluster some service points for real-world postal services' instances, as the TW density is sparse. Starting from the original network G, we create a clustered network, i.e., a network which contains fewer nodes and arcs. Each node in the clustered network represents one or several nodes in V. The depot nodes p and q and the customer nodes with TWs are never clustered with other nodes. The proposed clustering algorithm is iterative. At each iteration, it clusters a pair of nodes without TWs (which may represent several original nodes) to create a new node associated with the location of one of the nodes it represents. The traveling time matrix is then updated after computing a Hamiltonian path passing through all the nodes represented by the new node.

A pseudo-code of the clustering algorithm is given in Algorithm 1. The following notation is used. First, we number the nodes in V from 0 to $|V|-1$ and associate nodes p and q to numbers 0 and $|V|-1$. Thus, the customer nodes are numbered from 1 to $|V|-2$. Let $J = \{1, \ldots, |V|-2\}$ and $\tilde{J} \subset J$ be the nodes of the customers without a TW. At each iteration i of the algorithm, the clustered problem is represented by the following vectors and matrix.

N_i: Vector of dimension $|V|-2$ of subsets of customer nodes. For all $j \in J$, $N_i(j)$ is the (possibly empty) subset of customers that are represented by customer j. This vector has the following properties: all customers with a TW is represented by itself ($N_i(j) = \{j\}, \forall j \in J \setminus \tilde{J}$); every customer must belong to one subset ($\bigcup_{j \in J} N_i(j) = J$); and a customer cannot belong to two different subsets ($N_i(j_1) \cap N_i(j_2) = \emptyset, \forall (j_1, j_2) \in J^2$ such that $j_1 \neq j_2$).

S_i: Vector of dimension $|V|-2$ of total service times. For all $j \in J$, $S_i(j) = \sum_{v \in N_i(j)} s_v$ if

Algorithm 1 Pseudo-code of the clustering phase

Initialize N_0, S_0, ITT_0, and M_0
$(k_1, l_1) \leftarrow \text{argmin}_{(k,l)\in\tilde{J}^2} M_0(k,l)$
$d_1 \leftarrow M_0(k_1, l_1)$
$i \leftarrow 1$
while $d_i \leq d^{max}$ **do**
    $\mathcal{C}_i \leftarrow N_{i-1}(k_i) \cup N_{i-1}(l_i)$
    $N_i \leftarrow N_{i-1}$, $S_i \leftarrow S_{i-1}$, $ITT_i \leftarrow ITT_{i-1}$, and $M_i \leftarrow M_{i-1}$
    if $d_i = 0$ **then**
        $N_i(k_i) \leftarrow \mathcal{C}_i$ and $N_i(l_i) \leftarrow \emptyset$
        $S_i(k_i) \leftarrow S_i(k_i) + S_i(l_i)$
        $M_i(j, l_i) = M_i(l_i, j) = \Delta, \quad \forall j \in J$
    else
        $p'_i \leftarrow \text{argmin}_{v\in V\setminus\mathcal{C}_i} \sum_{j\in\mathcal{C}_i} t_{vj}$ and $q'_i \leftarrow \text{argmin}_{v\in V\setminus(\mathcal{C}_i\cup\{p'_i\})} \sum_{j\in\mathcal{C}_i} t_{vj}$
        if $D_{q'_i} < R_{p'_i}$ **then**
            $(p'_i, q'_i) \leftarrow (q'_i, p'_i)$
        end if
        Compute a shortest Hamiltonian path $(p'_i, v_i^1, \ldots, v_i^{|\mathcal{C}_i|}, q'_i)$ between p'_i and q'_i and passing through all nodes $v_i^1, \ldots, v_i^{|\mathcal{C}_i|}$ in $\mathcal{C}_i$
        $N_i(v_i^1) \leftarrow \mathcal{C}_i$ and $N_i(j) \leftarrow \emptyset, \quad \forall j \in \mathcal{C}_i \setminus \{v_i^1\}$
        $S_i(v_i^1) \leftarrow S_i(k_i) + S_i(l_i)$
        $ITT_i(v_i^1) \leftarrow \sum_{j=1}^{|\mathcal{C}_i|-1} t_{v_i^j v_i^{j+1}}$ and $ITT_i(j) \leftarrow 0, \quad \forall j \in \mathcal{C}_i \setminus \{v_i^1\}$
        if $k_i \neq v_i^1$ **then**
            $M_i(j, k_i) = M_i(k_i, j) = \Delta, \quad \forall j \in J$
        end if
        if $l_i \neq v_i^1$ **then**
            $M_i(j, l_i) = M_i(l_i, j) = \Delta, \quad \forall j \in J$
        end if
        for $j \in V \setminus \mathcal{C}_i$ such that $N_i(j) \neq \emptyset$ **do**
            $M_i(v_i^1, j) \leftarrow ITT_i(v_i^1) + t_{v_i^{|\mathcal{C}_i|}, j}$
            $M_i(j, v_i^1) \leftarrow ITT_i(j) + t_{j, v_i^1}$
        end for
    end if
    $i \leftarrow i + 1$
    $(k_i, l_i) \leftarrow \text{argmin}_{(k,l)\in\tilde{J}^2} M_{i-1}(k,l)$
    $d_i \leftarrow M_{i-1}(k_i, l_i)$
end while

$N_i(j) \neq \emptyset$. Otherwise, the value of $S_i(j)$ is irrelevant.

ITT_i**:** Vector of dimension $|V| - 2$ of the node internal traveling times. For all $j \in J$, $ITT_i(j) = 0$ if $|N_i(j)| \leq 1$. Otherwise, $ITT_i(j)$ approximates the minimal travel time required to visit all customers in $N_i(j)$.

M_i: Matrix of dimension $|V|-2 \times |V|-2$ of the arc traveling times. For all $(j_1, j_2) \in J^2$, $M_i(j_1, j_2) = \Delta$ if $j_1 = j_2$, $N_i(j_1) = \emptyset$ or $N_i(j_2) = \emptyset$, where Δ is a constant larger than the maximum distance between two nodes. Otherwise, $M_i(j_1, j_2)$ approximates the traveling time from j_1 to j_2, including the internal traveling time at j_1.

The algorithm starts by initializing these vectors and matrix (Step 1) as follows. For all $j \in J$, $N_0(j) = j$, $S_0(j) = s_j$, and $ITT_0(j) = 0$, which means that every node in N_0 represents a single customer, with its own service time, and no internal travel time, as no clustering has been applied yet. For all $(j_1, j_2) \in J^2$, $M_0(j_1, j_2) = \Delta$ if $j_1 = j_2$, to prevent the selection of (j, j) as a good candidate for clustering. Otherwise, $M_0(j_1, j_2) = t_{j_1 j_2}$. In Step 2, the algorithm identifies in matrix M_0 the pair of nodes $(k_1, l_1) \in \tilde{J}^2$ yielding the minimal traveling time, then stores it in d_1 (Step 3). It then sets the iteration counter i to 1 (Step 4). As long as d_i does not exceed a predetermined maximum time d^{max} (Step 5), the selected nodes k_i and l_i will be merged in iteration i, resulting in a node containing the customer nodes in $\mathcal{C}_i$, *i.e.,* the ones already included in $N_{i-1}(k_i)$ and $N_{i-1}(l_i)$ (Step 6). It starts by making copies of the vectors and matrix N_{i-1}, S_{i-1}, ITT_{i-1} and M_{i-1} (Step 7). These copies will be updated to define the clustered problem at iteration i.

Two cases can occur. If $d_i = 0$ (test at Step 8), a case that may happen when several customers are located in the same building, all customers represented by k_i and l_i are arbitrarily assigned to k_i and all information related to l_i become obsolete (Steps 9 - 11). Note that, if $d_i = 0$, the traveling time between any pair of nodes in $\mathcal{C}_i$ is equal to 0 because in all previous iterations i', $d_{i'}$ was also equal to 0. If $d_i > 0$, the merging process is more complex (Steps 12-31). Given that, in the current clustered problem, the customers in $\mathcal{C}_i$ will be visited consecutively, we would like to approximate the minimal traveling time ITT required to visit the customers in $\mathcal{C}_i$. To do so, we first select two nodes p'_i and q'_i not in $\mathcal{C}_i$ that might immediately precede and succeed to the node representing $\mathcal{C}_i$ in the solution. We choose these nodes as the two closest neighbours on average to all nodes in $\mathcal{C}_i$ (Step 13). If they have TWs and are badly ordered, we switch them in Step 15. We then compute in Step 17 a shortest Hamiltonian path that starts in p'_i, ends in q'_i and visits all nodes in $\mathcal{C}_i$. ITT is then set to the total traveling time between the first node v_i^1 after p'_i and the last node $v_i^{|\mathcal{C}_i|}$ before q'_i in this path, i.e, $ITT = \sum_{j=1}^{|\mathcal{C}_i|-1} t_{v_i^j v_i^{j+1}}$. Given that $|\mathcal{C}_i|$ is relatively small (less than 25 in our tests) and none of the customers in $\mathcal{C}_i$ has a TW, these shortest paths can be computed rapidly with a general-purpose MIP solver. The customers in $\mathcal{C}_i$ are then assigned to node v_i^1 in Step 18. Some entries of the traveling time matrix M_i are updated in Steps 21 to 29. Note that, in this last step (Step 29), we could use t_{j', v_i^1} instead of t_{j, v_i^1} where j' is the second to last node in a shortest Hamiltonian path starting at j, finishing at v_i^1 and going through all nodes in $N_i(j)$. But, given that j and j' are typically very close to each other

when $j' \in N_i(j)$, we have decided to use t_{j,v_i^1} as a proxy for t_{j',v_i^1}. The current iteration ends by increasing the iteration counter and selecting in Step 33 the next pair of nodes to cluster.

Once the algorithm terminates, the clustered problem is obtained by creating one node for each non-empty subset $N_{i-1}(j)$, $j \in J$, using the corresponding service times in vector S_{i-1} and the corresponding traveling times in matrix M_{i-1}. The vector ITT_{i-1} is not required anymore as the internal traveling times are included in the traveling times provided in matrix M_{i-1}.

Algorithm 1 stops when the traveling time d_i between the next pair of nodes (k_i, l_i) to cluster exceeds a predetermined threshold as clustering these two nodes seems riskier. Indeed, in preliminary tests, we observed that being too aggressive in the clustering phase may have significant drawbacks as too many approximations are introduced especially when updating the traveling time matrix. To avoid this, we also tested two other criterion. The first is to stop clustering when the number of nodes in the clustered problem reached a prefixed minimum number. The second is local and forbids clustering too many original nodes into a single one, i.e., a maximal cardinality n^{max} on each subset $N_i(j)$, $j \in J$, is imposed. When the latter option is activated, we filtered the nodes allowed to be clustered at step 33 by restricting to arcs $(k, l) \in \tilde{J}^2 \mid |N_i(k)| + |N_i(l)| \leq n^{max}$ in the argmin's scope.

4.3.2 Solution of the clustered problem

After the clustering, we solve the clustered problem using our CP model (Section 4.2.1) with the goal of minimizing RD. Even if the clustered problem is smaller than the original one, there are still some TWs, and the problem remains complex. Nevertheless, for clustered problem instances with around 100 nodes or fewer, high-quality solutions can be found in a reasonable time.

4.3.3 Disaggregation

After solving the clustered problem to find a clustered solution denoted by *Seq*, we compute a solution to the original problem, respecting the sequence of *Seq* to a certain extent. This solution is found by solving the original problem subject to additional precedence constraints derived from *Seq* and presented below. Two algorithms, described afterwards, can be used to compute this solution.

4.3.3.1 Precedence constraints

Let us partition the nodes in *Seq* (except the depot nodes) in two subsets K and W. Subset $K = \{K_1, \ldots, K_{n^K}\}$ contains the n^K nodes without a TW, hereafter called the clustered nodes even if such a node corresponds to a single original node. Subset $W = \{W_1, \ldots, W_{n^W}\}$ contains the n^W nodes with a TW (they all remained unclustered). Node K_i is the i^{th} visited clustered node, but not necessarily the i^{th} node of *Seq* if some nodes of W are visited earlier. Let j^- (resp. j^+) be the index i of the closest clustered node K_i before (resp. after) W_j in *Seq*. We assume that $i = 0$ if W_j is the first node visited in *Seq* and that $i = n^K + 1$ if W_j is the last one visited.

Let $\theta \geq 1$ be an integer flexibility parameter that indicates how many neighboring clustered nodes may be merged. The precedence constraints are divided into two groups, depending on the types of nodes involved.

— For pairs of clustered nodes, the following constraints allow rearrangement of the original nodes, respecting *Seq* with flexibility θ:

$$a \prec b, \quad \forall i \in \{1, \ldots, n^K - \theta\}, \quad \forall a \in \mathcal{C}_{K_i}, \quad \forall b \in \mathcal{C}_{K_{i+\theta}} \tag{4.11}$$

where, as in Section 4.2, $a \prec b$ means that a must be visited before b. When $\theta = 1$, the nodes within each clustered node can be rearranged but they must respect their order with respect to the nodes in the preceding and succeeding clustered nodes.

— The nodes with TWs are now allowed to merge with their θ closest clustered neighbors:

$$b \prec W_j, \quad \forall\, W_j \in W \mid j^- - \theta \geq 1, \quad \forall b \in \mathcal{C}_{K_{(j^- - \theta)}} \tag{4.12}$$

$$a \succ W_j, \quad \forall\, W_j \in W \mid j^+ + \theta \leq n^K, \quad \forall a \in \mathcal{C}_{K_{(j^+ + \theta)}} \tag{4.13}$$

If $\theta = 1$, every node W_j can be reordered with the nodes in its immediate predecessor and successor clustered node to allow a better positioning of this time constrained node.

As an example, let $Seq = \{p - \hat{A} - \hat{B} - W_1 - \hat{C} - W_2 - \hat{D} - q\}$ be the solution of the clustered problem, where $K = \{\hat{A}, \hat{B}, \hat{C}, \hat{D}\}$ is the clustered node set and W_1 and W_2 are nodes with TWs. Let $\theta = 1$.

The precedence constraints are:

$$a \prec b, \quad \forall a \in \mathcal{C}_{\hat{A}}, \quad \forall b \in \mathcal{C}_{\hat{B}} \tag{4.14}$$

$$b \prec c, \quad \forall b \in \mathcal{C}_{\hat{B}}, \quad \forall c \in \mathcal{C}_{\hat{C}} \tag{4.15}$$

$$c \prec d, \quad \forall c \in \mathcal{C}_{\hat{C}}, \quad \forall d \in \mathcal{C}_{\hat{D}} \tag{4.16}$$

for the clustered nodes, and

$$W_1 \succ a, \quad \forall a \in \mathcal{C}_{\hat{A}} \tag{4.17}$$
$$W_1 \prec d, \quad \forall d \in \mathcal{C}_{\hat{D}} \tag{4.18}$$
$$W_2 \succ b, \quad \forall b \in \mathcal{C}_{\hat{B}} \tag{4.19}$$

for the nodes with TWs.

For more flexibility, θ can be increased. In our example, with $\theta = 2$, (4.14)–(4.16) are replaced by

$$a \prec c, \quad \forall a \in \mathcal{C}_{\hat{A}}, \quad \forall c \in \mathcal{C}_{\hat{C}} \tag{4.20}$$
$$b \prec d, \quad \forall b \in \mathcal{C}_{\hat{B}}, \quad \forall d \in \mathcal{C}_{\hat{D}} \tag{4.21}$$

and (4.17)–(4.19) become

$$W_2 \succ a, \quad \forall a \in \mathcal{C}_{\hat{A}}. \tag{4.22}$$

These additional precedence constraints increase the precedence lists $V^+(i)$, $i \in V$, defined in Section 4.2. They are, therefore, imposed in the CP model through constraints (4.8) and in the TBF by eliminating arcs in set A. Note that no precedence constraints are imposed between two nodes in W, giving the opportunity to change their order of visit if this change respects the TWs and the precedence relationships.

Taking into account the precedence constraints, the problem can be solved using one of the two methods described in Sections 4.3.3.2 and 4.3.3.3.

4.3.3.2 Pure CP disaggregation algorithm

To compute a final disaggregated route, we can resort to a pure CP algorithm which simply solves the CP-WOF model augmented with all precedence constraints Stated above. To speed up this algorithm, we tighten the TWs on the depot nodes p and q using the upper bound on the optimal RD provided by the value $\bar{z}_{RD}$ of the solution found for the clustered problem. More precisely, the TW at node p can be replaced by $[\max\{R_p, R_q - \bar{z}_{RD}\}, D_p]$ and that at node q by $[R_q, \min\{D_p + \bar{z}_{RD}, D_q\}]$, where we assume here that $[R_p, D_p]$ and $[R_q, D_q]$ are the TWs resulting from the preprocessing procedure mentioned in Section 4.2. Obviously, if one of these TWs is reduced in this way, preprocessing can be applied again.

4.3.3.3 Mixed CP and TBF disaggregation algorithm

Given that the TBF-based branch-and-cut algorithm is not efficient at minimizing the RD while the CP algorithm is, we propose an alternative disaggregation algorithm that combines both algorithms. This algorithm is as follows.

Step Dis-1: Minimize RD via CP to obtain a good upper bound for Step Dis-2.

Step Dis-2: Minimize TT via the TBF after imposing this upper bound on the RD.

Step Dis-3: Minimize RD via CP subject to the sequence computed in Step Dis-2.

In Step Dis-1, the CP algorithm is ran for only 1 second to solve the CP-model with the additional precedence constraints derived from the clustered solution found and by setting $\theta = 1$. Indeed, preliminary experiments (see Appendix 4.B) have shown that this algorithm finds very rapidly a high-quality solution but struggles to improve it afterwards or prove its optimality. Moreover, increasing the value of θ makes it difficult to compute a good-quality solution very rapidly. In Step Dis-2, the TBF also includes the precedence constraints arising from the clustered solution. However, the efficiency of the TBF-based algorithm to minimize the TT when the number of precedence constraints is sufficiently large allows to set θ to a slightly larger value (*e.g.*, $\theta = 2$ or 3), providing more flexibility in this step. In fact, to increase its efficiency, an artificial TW derived from the visiting times T_i of the clustered nodes $K_i \in K$ in the clustered solution is created for every service point that was not originally time-constrained. For node $a \in \mathcal{C}_{K_i}$, the artificial TW is $[T_{i-\theta}, T_{i+1+\theta}]$ because a node in $\mathcal{C}_{K_i}$ should be visited after the first node in $\mathcal{C}_{K_{i-\theta}}$ and before the last one in $\mathcal{C}_{K_{i+\theta}}$. These artificial TWs are only used to reduce the number of variables in the TBF; they are not taken into consideration when identifying infeasible paths. Finally, Step Dis-3 simply ensures that all avoidable idle time is removed from the computed route.

4.4 Results on small and medium-sized benchmark instances

In this section, we consider small and medium-sized instances from the literature with dense TWs to motivate the use of a bi-objective cost function in the TSPTW considered. Given the size of these instances, we solved them only with the CP solver. For our tests, we used only one set of well-known instances, namely, that of [67], hereafter called the GHLS instances. They are grouped in classes denoted nXXwYY, where XX is the number of nodes (between 20 and 100) and YY indicates the width of the TWs. There are 21 classes and 5 instances per class, for a total of 105 instances. Considering the size of these instances and the fact that all nodes have a TW, no clustering is applied for these instances. Note that we also

performed tests on other instance sets, including that of [74] and [78], and obtained similar results.

Our computational experiments were ran on an Intel(R) Xeon(R) X5675 at 3.07 GHz using a single thread. The code was compiled in C++ and uses IBM CPLEX CP-Optimizer 12.7.1.0. Since the TSPTW may be solved many times in a CFRS procedure, we imposed a tight time limit of 60 seconds. For the WOF model, we used $\gamma^{TT} = 8$ and $\gamma^{RD} = 20$ except for the parameter sensitivity analysis. Furthermore, we define z^{WOF}, the WOF value of a solution, as $z^{WOF} = 8z^{TT} + 20z^{RD}$, where z^{TT} and z^{RD} are the TT and RD of this solution, respectively.

4.4.1 Comparative results

Figures 4.2 and 4.3 compare the solutions obtained by minimizing the WOF (first approach) versus those produced by minimizing TT or RD only (the reference approach), respectively. The figures show, for each instance class, the average savings (in percentage and represented by the dots) on the WOF value obtained by the solutions of the first approach over those of the reference one. A negative saving means that, on average, the reference approach yields a solution with a lower WOF value. Furthermore, the histograms indicate, for each class, the average differences in TT and in RD between these solutions. For example, in Figure 4.2, the bars represent the average variations in TT (blue) and in RD (red) between the solutions obtained with the WOF and when minimizing TT only.

From these results, we make the following observations. Compared to the TT solutions (Figure 4.2), the WOF solutions yield savings for an instance class ranging from -0.24% to 6.76%, and an average global saving of 2.53%. Figure 4.2 shows that the increase in the TT is usually compensated for by the reduction in the RD, and so the delivery companies could have routes with shorter durations while lowering their routing costs. This is not the case for the instance group n100w80, where TT-optimal solutions already have a good RD value, and the tight time limit does not allow the solver to generate such good solutions for the WOF (even if the average cost increases only by 0.24% for this group). Figure 4.3 shows that when there are many TWs, even if they are relatively wide, it is too costly to minimize only the RD. Indeed, in most cases, the WOF solutions exhibit a small increase in RD but a large decrease in TT compared to the solutions obtained by minimizing only RD. This leads to savings ranging from -0.09% to 4.63%, with a global average of 2.52%.

Both series of results clearly highlight that focusing on a single objective can produce solutions that are significantly sub-optimal when both criteria play an important role in the routing costs. Therefore, a bi-objective function should be considered during optimization.

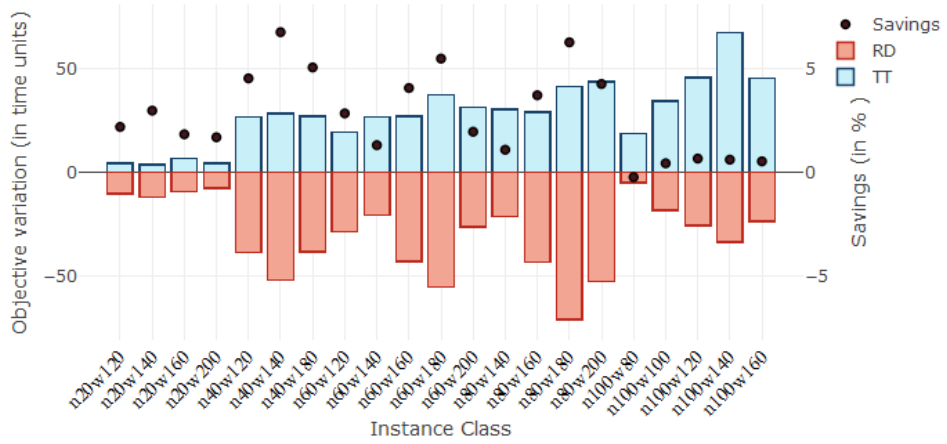


Figure 4.2 Savings and average TT and RD variations per class, for WOF versus TT only.

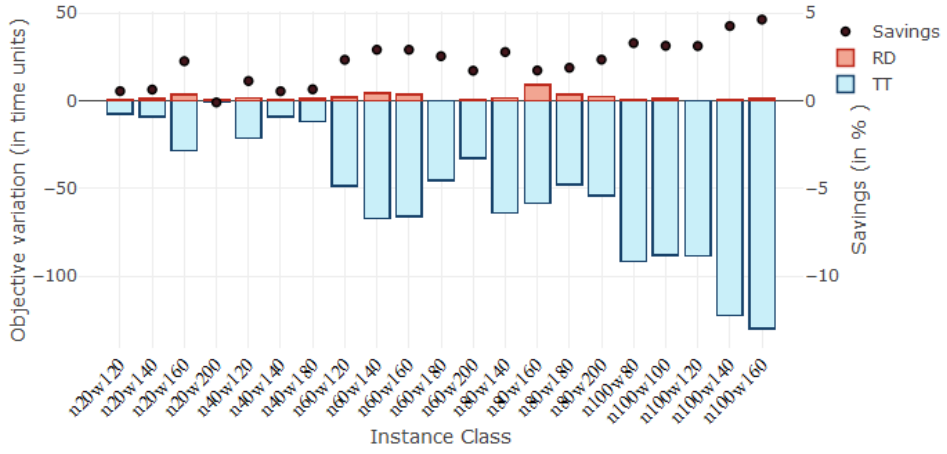


Figure 4.3 Savings and average TT and RD variations per class, for WOF versus RD only.

4.4.2 Sensitivity analysis on the weights of the WOF

Results for different ratios of the parameters γ^{TT} and γ^{RD} are presented in Table 4.1. For each setting, we report average results computed over the solutions obtained for all GHLS instances. In this table, the first column specifies the values of γ^{TT} and γ^{RD} used in the WOF model. The next three columns provide the average TT and RD values of the solutions, as well as the average computational time in seconds. Then, the last two columns report the average z^{WOF} savings in percentage with respect to the solutions computed when minimizing TT only and RD only, respectively. In the last two lines, we provide as references the average TT and RD values of the solutions computed when minimizing TT only and RD only.

This table first shows that it is always relevant to consider a bi-objective function because savings are always positive for the tested ratios. Furthermore, the lower the ratio γ^{TT}/γ^{RD} is, the closer the RD is to optimality, which is not surprising as a larger proportion of the cost is dedicated to minimizing RD, and the CP algorithm performs well on this part of the objective function. Finally, we observe that the average computational time increases as the ratio γ^{TT}/γ^{RD} converges to one, indicating that it is difficult to obtain a perfect tradeoff between TT and RD.

Table 4.1 Comparing the WOF solutions with different parameter settings

γ^{TT}/γ^{RD}	TT	RD	Time (s)	z^{WOF} savings (%)	
				vs TT-only	vs RD-only
1/20	475.9	539.9	46.8	5.07%	0.34%
5/20	470.2	541.0	52.3	3.43%	1.69%
8/20	469.1	541.5	56.5	2.53%	2.52%
13/20	466.1	543.3	58.5	1.39%	3.64%
20/20	463.5	544.5	59.0	0.43%	4.96%
TT-Only	440.6	571.8			
RD-Only	521.1	539.6			

4.5 Results for large industrial instances with sparse TWs

We have studied two instances, labelled *a* and *b*, provided by our industrial partner, Giro. They commercialize the optimization software GeoRoute which is used by some of the largest postal organizations worldwide. These instances have 475 (Full_a) and 458 (Full_b) nodes and just a few TWs (resp. 6 and 8), which is an extreme case of flexibility. We also generated instances with 200 and 300 nodes, selected randomly from those available, retaining all of the nodes with TWs. We refer to the instances as rd.XXX.Yg where XXX is the number of nodes

selected from the original instance Full_g (where g = a or b) and Y is an identifier. For our tests, we used the same computer as described above, still with a single thread. In addition to CP-Optimizer, the disaggregation phase using TBF relied on the MIP solver Cplex, version 12.7.1.0. Note that, in the original instances, the total service duration accounts for more than 50% of the total route duration, and this time is incompressible.

We present in this section several results about our clustering approach described in Section 4.3. First, we compare the usage of the two algorithms presented in Sections 4.3.3.2 and 4.3.3.3 for the disaggregation phase (Section 4.5.1). Then, we perform a sensitivity analysis on the value of some parameters used in the disaggregation phase (Section 4.5.2) and the aggregation phase (Section 4.5.3). Finally, we compare the performance of the clustering approach and the direct CP-WOF approach (Section 4.5.4) and provide optimality gaps for the solutions computed by the clustering algorithm (Section 4.5.5).

In the following tables, $\#N$ denotes the number of nodes remaining in the clustered problem, z^{WOF}_{Clus} the WOF value (cost) of the solution computed for the clustered problem, z^{WOF}_{Fin} the cost of the final solution and $Time$ the total computation time in seconds. For each instance, the best cost is highlighted in bold. Upon equality, only the cost obtained with fastest method is emboldened. Note that the WOF value of a solution is defined as in the previous section, i.e., $z^{WOF} = 8z^{TT} + 20z^{RD}$.

4.5.1 Comparison of disaggregation algorithms

Sections 4.3.3.2 and 4.3.3.3 describe two algorithms for disaggregating a clustered solution. The first solves a CP model whereas the second combines this CP model with a TBF. In Table 4.2, we report the results obtained with both algorithms when a maximum of 20 seconds is allocated to compute a solution in each phase (the total time may exceed 40 seconds because of the time required to cluster the nodes and build the models) and the parameters are set as follows: $d^{max} = 20$, $n^{max} = 20$, and $\theta = 1$.

From these results, we observe first that the costs of the final solutions are much less than those of the clustered solutions. In general, the final solutions computed by both disaggregation algorithms are of similar quality, with a small advantage for the mixed CP and TBF algorithm. On the other hand, this algorithm is much faster than the pure CP algorithm, with an average total time reduction of 30% (obtained by reducing the disaggregation time by 70% on average). Given that most times required by the mixed CP and TBF algorithm are below the 40-second time limit, we conclude that the corresponding solutions computed by this algorithm are optimal for the disaggregation phase. Moreover, with the pure CP algorithm, increasing θ to 2 or more yields poor-quality solutions in a limited computational

time. These observations led us to use the mixed CP and TBF disaggregation algorithm to produce the subsequent results.

Table 4.2 Comparative results for the disaggregation algorithms

			Pure CP		CP + TBF	
Instance	#N	z_{Clus}^{WOF}	z_{Fin}^{WOF}	Time (s)	z_{Fin}^{WOF}	Time (s)
rd.200.1a	75	152.63	151.05	41.5	**151.05**	23.4
rd.200.2a	69	164.55	162.45	41.3	**162.45**	23.2
rd.200.3a	72	162.08	159.47	41.4	**159.47**	23.2
rd.300.1a	93	188.51	185.13	42.6	**184.87**	26.9
rd.300.2a	86	185.50	181.35	42.7	**181.26**	26.7
rd.300.3a	89	189.65	186.45	42.8	**186.10**	26.9
Full_a	94	232.35	226.78	48.0	**226.75**	42.9
rd.200.1b	79	177.89	176.42	41.3	**176.42**	22.9
rd.200.2b	75	174.10	172.19	41.4	**172.19**	23.1
rd.200.3b	70	166.26	165.23	41.6	**165.23**	22.9
rd.300.1b	84	207.86	**204.54**	42.8	204.73	41.9
rd.300.2b	84	206.25	204.27	42.8	**203.78**	26.9
rd.300.3b	88	208.49	205.23	42.6	**205.06**	26.7
Full_b	95	252.31	247.55	47.5	**247.51**	41.7
Avg	82	190.60	187.72	42.9	**187.63**	28.5

4.5.2 Impact of the flexibility parameter value

In this section, we assess the impact of the value of flexibility parameter θ on the solution quality and the total computational time. This parameter controls the number of clustered nodes that can be reordered around each node in the disaggregation phase. In Table 4.3, we present the results obtained for three different values of θ, namely, $\theta = 1$ as in the previous section, as well as $\theta = 2$ and $\theta = 3$ which allow more flexibility during disaggregation. For these tests, a maximum time limit of 300 seconds for the whole solution process was set to yield optimal solutions for the disaggregation phase in all instances except for two with $\theta = 3$. From these results, we observe that increasing the flexibility enables slightly improving the quality of the final solution (by 0.13% and 0.18% for $\theta = 2$ and $\theta = 3$, respectively) but could be costly in time consumption. Because $\theta = 2$ offers a good trade-off between solution quality and computational time, we use this value in the following sections.

4.5.3 Impact of the clustering parameter values

As mentioned above, all our basic tests were run using $d^{max} = 20$ and $n^{max} = 20$, yielding clustered problems with an average of 82 clustered nodes and a maximum of 95 nodes. To

Table 4.3 Comparative results for three values of θ

			$\theta = 1$		$\theta = 2$		$\theta = 3$	
Instance	#N	z_{Clus}^{WOF}	z_{Fin}^{WOF}	Time (s)	z_{Fin}^{WOF}	Time (s)	z_{Fin}^{WOF}	Time (s)
rd.200.1a	75	152.63	151.05	23.4	**150.72**	25.1	150.72	29.2
rd.200.2a	69	164.55	162.45	23.2	**162.26**	25.7	162.26	34.2
rd.200.3a	72	162.08	159.47	23.2	159.28	24.2	**159.02**	42.3
rd.300.1a	93	188.51	184.87	26.9	184.24	35.5	**183.91**	76.9
rd.300.2a	86	185.50	181.26	26.7	181.02	59.6	**180.95**	95.3
rd.300.3a	89	189.65	186.10	26.9	186.13	57.0	**185.31**	102.4
Full_a	94	232.35	226.75	42.9	**226.52**	71.7	226.92	300.0
rd.200.1b	79	177.89	176.42	22.9	176.42	24.2	**176.39**	30.1
rd.200.2b	75	174.10	172.19	23.1	172.10	24.6	**171.82**	32.1
rd.200.3b	70	166.26	165.23	22.9	165.23	24.3	**165.09**	28.4
rd.300.1b	84	207.86	204.54	65.8	204.05	35.4	**203.98**	80.0
rd.300.2b	84	206.25	203.78	26.9	**203.52**	44.7	203.52	164.8
rd.300.3b	88	208.49	205.06	26.7	204.57	37.3	**204.20**	115.9
Full_b	95	252.31	247.51	41.7	**246.97**	78.8	247.65	300.0
Avg	82	190.60	187.62	30.2	187.36	40.6	**187.27**	102.3

show the impact of using clustered problems that are less clustered, we solved all instances using a less aggressive clustering procedure. This procedure stopped when the number of clustered nodes reached 100 (resp. 120) nodes, except when the minimum traveling time d_i is equal to the previous one. In this case, the aggregation continues until finding a larger minimum traveling time, yielding a clustered problem with slightly less than 100 (resp. 120) nodes. The results of these tests are reported in Table 4.4 for the three different stopping criterion, where the latter are denoted $\#N \approx 100$ and $\#N \approx 120$.

These results show that the second approach ($\#N \approx 100$) produces better solutions on average than the other two. The third one ($\#N \approx 120$) is the fastest. It may be explained by the fact that the clustered nodes contain less customers and, thus, the disaggregation problem is easier to solve, but less permissive. The first approach ($d^{max} = 20$, $n^{max} = 20$) has the most aggressive clustering phase, increasing the probability of sub-optimal inner routes in the cluster nodes and a poorer approximation. This could be counter-balanced by the fact that the clustered problem is easier to solve, due to fewer nodes in the instances. For the following section, the results were produced using the clustering setting $\#N \approx 100$.

4.5.4 Direct CP-WOF algorithm versus clustering algorithm

In this section, we compare the direct CP-WOF algorithm of Section 4.2 and the clustering heuristic. For the latter, we imposed a time limit of 20 seconds for solving the clustered

Table 4.4 Comparative results for three clustering procedure stopping criteria

	$d^{max} = 20$, $n^{max} = 20$			$\#N \approx 100$			$\#N \approx 120$		
Instance	#N	z_{Fin}^{WOF}	Time	#N	z_{Fin}^{WOF}	Time	#N	z_{Fin}^{WOF}	Time
rd.200.1a	75	**150.72**	25.7	100	151.23	24.2	117	151.02	23.6
rd.200.2a	69	**162.26**	26.2	100	162.33	24.4	113	162.33	23.8
rd.200.3a	72	159.28	24.7	100	**159.16**	24.0	104	159.44	24.0
rd.300.1a	93	184.24	36.1	94	**183.33**	34.7	119	185.92	31.1
rd.300.2a	86	**181.02**	58.7	99	181.23	35.8	119	181.49	28.6
rd.300.3a	89	185.47	70.3	100	185.78	32.2	115	**184.75**	31.6
Full_a	94	226.52	76.4	99	**225.87**	94.7	120	227.83	63.2
rd.200.1b	79	176.42	24.8	100	**173.94**	24.3	115	173.99	23.5
rd.200.2b	75	172.10	25.4	92	172.17	24.2	110	**171.49**	23.7
rd.200.3b	70	165.23	25.1	94	**164.20**	24.4	105	164.81	23.7
rd.300.1b	84	**204.05**	35.9	98	204.78	34.2	118	205.76	28.3
rd.300.2b	84	**203.52**	45.3	100	203.68	30.9	120	204.10	28.8
rd.300.3b	88	**204.57**	37.9	100	205.44	32.2	120	206.72	29.8
Full_b	95	246.97	80.1	100	**244.78**	93.6	120	249.98	59.3
Avg	82.4	187.31	42.3	98.3	**186.99**	38.1	115.4	187.83	31.7

problem. For both algorithms, a 60-second overall time limit was enforced.

The results are reported in Table 4.5 where the last column (Imp) gives the cost improvement in percentage obtained by the solutions produced by the clustering heuristic over those computed by the direct CP-WOF algorithm. On average, the clustering heuristic yields an average cost improvement of 1.60%, in around half the time allowed for the CP-WOF method. Note that the disaggregation method terminated before the time limit for all the instances except the full ones, ensuring that there is no better feasible disaggregation given the computed clusters and clustered sequence. Observe also that the cost z_{Clus}^{WOF} of the clustering phase solution is, on average, less than that of the direct CP-WOF algorithm solution (z^{WOF}).

Figure 4.4 presents the cost of the best solution found in function of the computational time for the two original instances Full_a and Full_b and two different algorithms, namely, the direct CP-WOF algorithm (solid lines) and the proposed clustering algorithm (dashed-dotted lines). For the latter algorithm, the solution process stops when the disaggregation problem is solved to optimality at around 90 seconds. For these two instances, we clearly see that the clustering algorithm can provide much better quality solutions than the direct CP-WOF algorithm, even if a 60-second time limit was imposed.

Table 4.5 Comparative results between the CP-WOF algorithm and the clustering algorithm

	WOF		Clustering				
Instance	z^{WOF}	Time	#N	z^{WOF}_{Clus}	z^{WOF}_{Fin}	Time	Imp
rd.200.1a	157.28	60.0	100	151.96	**151.23**	23.4	3.37%
rd.200.2a	164.15	60.0	100	163.01	**162.33**	23.5	0.96%
rd.200.3a	159.30	60.0	100	160.54	**159.16**	23.5	0.08%
rd.300.1a	185.55	60.0	94	187.20	**183.33**	34.3	1.01%
rd.300.2a	186.18	60.0	99	184.24	**181.23**	35.4	2.26%
rd.300.3a	186.03	60.0	100	188.60	**185.78**	31.5	0.12%
Full_a	234.90	60.0	99	231.70	**226.43**	60.0	3.17%
rd.200.1b	176.84	60.0	100	175.32	**173.94**	23.6	1.41%
rd.200.2b	173.15	60.0	92	173.50	**172.17**	23.7	0.49%
rd.200.3b	166.42	60.0	94	165.23	**164.20**	23.9	1.16%
rd.300.1b	210.33	60.0	98	208.16	**204.78**	33.8	2.22%
rd.300.2b	206.44	60.0	100	205.64	**203.68**	30.3	1.12%
rd.300.3b	209.89	60.0	100	207.49	**205.44**	31.6	1.79%
Full_b	251.82	60.0	100	249.79	**245.15**	60.0	2.29%
Avg	190.59	60.0	98	189.46	**187.06**	32.7	1.60%

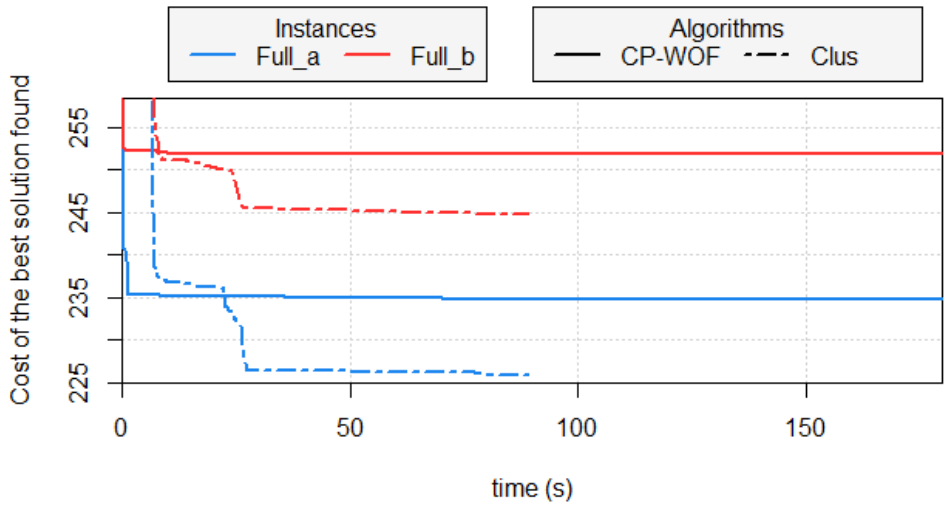


Figure 4.4 Cost of the best solution found in function of the computational time

4.5.5 Optimality gaps of the clustering algorithm solutions

To conclude the assessment of the quality of the solutions produced by the clustering algorithm, we computed for each instance a lower bound on its optimal value by solving the corresponding traveling salesman problem (TSP), defining the arc costs as the arc TTs only

(i.e., omitting the service times). Because $z^{WOF} = 8z^{TT} + 20z^{RD}$ and $z^{RD} \geq z^{TT} + ST$, where ST is the total service time, we deduce that $z^{WOF} \geq 28z^{TT} + 20ST$. Therefore, a valid lower bound (LB) on the optimal WOF value is given by $LB = 28z^{TT}_{TSP} + 20ST$, where z^{TT}_{TSP} is the optimal value of the corresponding TSP. For each instance, Table 4.6 reports this lower bound (LB) and the relative optimality gap between LB and z^{WOF}_{Fin}, the cost of solution computed by the clustering algorithm. The results indicate that the computed solutions are all within 1.8% of optimality, with an average of 1.3%. Given that the lower bounds are not necessarily tight (TWs are relaxed in the TSP), these gaps show the effectiveness of the proposed clustering algorithm.

Table 4.6 Optimality gaps

Instance	LB	z^{WOF}_{Fin}	Gap (%)
rd.200.1a	148.88	151.23	1.58
rd.200.2a	160.77	162.33	0.97
rd.200.3a	156.99	159.16	1.38
rd.300.1a	180.88	183.33	1.35
rd.300.2a	179.25	181.23	1.11
rd.300.3a	183.40	185.78	1.30
Full_a	223.37	226.43	1.37
rd.200.1b	171.96	173.94	1.15
rd.200.2b	169.13	172.17	1.79
rd.200.3b	161.99	164.20	1.37
rd.300.1b	202.68	204.78	1.04
rd.300.2b	200.77	203.68	1.45
rd.300.3b	201.89	205.44	1.76
Full_b	243.26	245.15	0.78
Avg	184.66	187.06	1.30

4.6 Conclusion

In this paper we highlighted the particularities of the well-known traveling salesman problem with time windows that arise in the context of postal operations, namely, the importance of minimizing not only the total travel time but also the total route duration, and the fact that very few customers have time windows, reducing significantly the efficiency of the standard TW preprocessing techniques. To solve this new TSPTW variant, we devised a three-step clustering heuristic which first clusters the unconstrained customers, then solves the clustered problem, before sequencing the customers inside the clustered nodes, with some flexibility between the consecutive nodes. Compared to a CP algorithm that tackles the problem at once, this heuristic is able to produce better-quality solutions (with an average optimality

gap of 1.3%) in shorter computational times on industrial instances involving up to 475 customers.

As a future work, we will tackle the postal territory design problem, considering that this design is a strategic decision which is taken approximately once per year perhaps, while optimizing the TSPTW occurs at an operational, daily, decision level.

4.A Appendix: Time Bucket Formulation

The TBF [75] is a MIP model which gathers several variables of a time-indexed formulation into time buckets, to reduce the combinatorial nature of the problem. Each bucket $b = [\underline{b}, \bar{b}]$ represents a portion of the time horizon. Let $\mathcal{B}$ be the set of all buckets used. With the directed graph $G = (V, A)$ introduced in Section 4.2, the TBF has three types of variables:

— $x_{ij} = 1$ if arc $(i, j) \in A$ is used; 0 otherwise.

— $z_i^b = 1$ if node i is visited in bucket b; 0 otherwise.

— $y_{ij}^b = 1$ if arc $(i, j) \in A$ is used and node i is visited in bucket b; 0 otherwise.

We define three subsets:

— $B_i \subseteq \mathcal{B}$ is the subset of buckets allowed for node i.

— $V^+(i) \subseteq V\backslash\{p\}$ (resp. $V^-(i) \subseteq V\backslash\{q\}$) is the subset of possible successors (resp. predecessors) of i. It can be tightly defined, as discussed in Section 4.2.

— $I_k(i, b)$ is the subset of B_k representing the potential starting buckets of node k if arc (k, i) is used and the visit to node i started in bucket b. We have

$$I_k(i, b_l) = \{b \in B_k : \bar{b}_{l-1} < \underline{b} + t_{ki} \leq \bar{b}_l\}$$

where $b_0 = -\infty$ is assumed.

Given this notation the time bucket relaxation (TBR) for the TSPTW is

$$\min \sum_{(i,j)\in A} t_{ij} x_{ij} \tag{4.23}$$

subject to:

$$\sum_{b \in B_i} z_i^b = 1, \qquad \forall i \in V \tag{4.24}$$

$$\sum_{j \in V^+(i)} y_{ij}^b = z_i^b, \qquad \forall i \in V \backslash \{q\}, \quad \forall b \in B_i \tag{4.25}$$

$$\sum_{k \in V^-(i)} \sum_{\beta \in I_k(i,b)} y_{ki}^\beta = z_i^b, \qquad \forall i \in V \backslash \{p\}, \quad \forall b \in B_i \tag{4.26}$$

$$\sum_{b \in B_i} y_{ij}^b = x_{ij}, \qquad \forall (i,j) \in A \tag{4.27}$$

$$x_{ij}, y_{ij}^b, z_i^b \in \{0,1\}, \qquad \forall i \in V, \quad \forall (i,j) \in A, \quad \forall b \in B_i. \tag{4.28}$$

The objective function (4.23) minimizes the TT. Constraints (4.24) ensure that each service point is visited in a single bucket. Constraints (4.25) and (4.26) link the variables y and z, while guaranteeing that a valid bucket is used via the set $I_k(i,b)$. Constraints (4.27) link x and y.

Model (4.23)–(4.28), as its name implies, is a relaxation of the TSPTW. The feasibility checks of the model are made as if every time was at the beginning of a bucket, so as shown in Figure 4.5, subtour elimination constraints (SECs) and infeasible path cuts (IPCs) should be added to obtain an exact formulation, the so-called TBF.

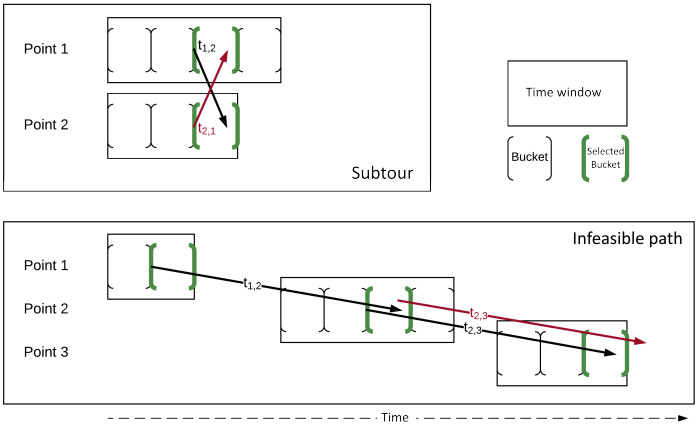


Figure 4.5 Subtour and infeasible path examples.

If the TT between two nodes is shorter than the width of a time bucket, subtours may occur. In dense areas or when the bucket width is large, the subtours may contain more than two nodes. In Figure 4.5, the infeasible path $(1[b_{1,2}], 2[b_{2,3}], 3[b_{3,3}])$, using respectively the second, third, and third buckets of nodes $1, 2$, and 3, is allowed by the TBR since the feasibility depends on $I_k(i,b)$, which selects buckets in B_k as long as a "time index" of the buckets

matches with one time index of b ($\in B_i$).

For every subtour detected, let $S \subset V\backslash\{p,q\}$ be the associated nodes. The following SEC must be added:

$$\sum_{(i,j)\in\delta(S)} x_{ij} \geq 1 \tag{4.29}$$

where $\delta(S)$ is the subset of arcs (i,j) of A such that $i \in S$ and $j \in V\backslash\{S\}$.

For every infeasible path $\mathcal{P}$ (which would violate some TWs), we add the IPC:

$$\sum_{(i,j)\in\mathcal{P}} x_{ij} \leq |\mathcal{P}| - 1. \tag{4.30}$$

The narrower the buckets, the more accurate the model and the fewer SECs and IPCs have to be added while solving the MIP. However, increasing the number of variables increases the combinatorial nature of the problem. With wider buckets, a node may not be assigned to the right bucket. However, if the solution remains feasible, we can retain it, because only the sequence matters. For this reason, the TBF may greatly reduce the TT.

We can try to capture the RD information of the sequence found, but it is not accurate, since we know only in which bucket the route finishes. Intuitively, the objective function may be modified to minimize z, where z is defined by the following constraint, leading to an overestimation of the real RD objective function:

$$z \geq \sum_{b\in B_q} \bar{b} z_q^b - \sum_{b\in B_p} \underline{b} z_p^b. \tag{4.31}$$

However, it is possible to be more accurate, especially when the waiting time is low, by replacing (4.31) by:

$$z \geq \sum_{b\in B_q} \underline{b} z_q^b - \sum_{b\in B_p} \bar{b} z_p^b \tag{4.32}$$

$$z \geq \sum_{(i,j)\in A} t_{ij} x_{ij}. \tag{4.33}$$

In both cases, the buckets have to be carefully assigned, and this requires the addition of some constraints; see [75]. Moreover, to obtain the exact RD, we must refine the buckets to increase the accuracy in terms of the departure time from and the arrival time at the depot. These two values greatly affect the computational time, and hence we do not use the TBF to minimize the RD. When an upper bound on RD is imposed like in Section 4.3.3.3, it is enforced through the addition of IPCs.

In the TBF, the bucket generation may be key to the performance. We generate the time buckets on a fixed base L so $b_0 = [0, L-1], b_1 = [L, 2L-1]$, and so on. If necessary, we replace the first and last bucket of each service point i by more accurate ones that match the TW. If $L = 10$ and for some i, $[R_i, D_i] = [27, 44]$, B_i is defined by $\{[27, 29], [30, 39], [40, 44]\}$ instead of $\{[20, 29], [30, 39], [40, 49]\}$. This allows us to discard some buckets in $I_k(i, b)$ that could have been considered with the original buckets.

The TBR is solved using Cplex 12.7.1.0, and SECs and IPCs are added via callbacks during the process. We use the *lazycallback* technology, i.e., we check for subtours and infeasible paths at every integer solution and then add the corresponding constraints (4.29) and (4.30) to the model.

4.B Appendix: Supporting Computational Results

In this appendix, we present some computational results to support our claims that the CP algorithm is efficient at minimizing RD for small and medium-sized instances while it is outperformed by the TBF-based algorithm when minimizing TT.

First, using the CP algorithm, we solved the 105 GHLS instances [67] with the objective of minimizing RD only. Figure 4.6 reports the performance profile curve for these experiments. It shows the number of instances solved to optimality (vertical axis) with respect to a time limit (horizontal axis) given in milliseconds and using a logarithmic scale. We can observe that 96 out of the 105 instances can be solved to optimality within one second of computational time and that the most difficult instance is solved in 71 seconds.

We conducted another set of experiments that consists of solving the 60 GHLS instances with less than 60 nodes and the objective of minimizing TT, using both the CP and the TBF-based algorithms. A maximum time of 300 seconds was imposed to solve each instance. In Figure 4.7, we report for each instance the optimality gaps obtained for the CP algorithm (blue) and the TBF-based algorithm (red) (some circles are superimposed). A cross indicates that the TBF-based algorithm cannot find a feasible solution for the corresponding instance within the time limit. These results show that the CP algorithm always returns a solution, whereas the TBF-based algorithm cannot for 19 instances. However, when the TBF-based algorithm can find a solution for an instance, it generally solves it to optimality. Table 4.7 reports the number of instances solved to optimality by both algorithms with respect to several time limits. From these results, we deduce that the TBF-based algorithm is much faster than the CP algorithm to prove optimality.

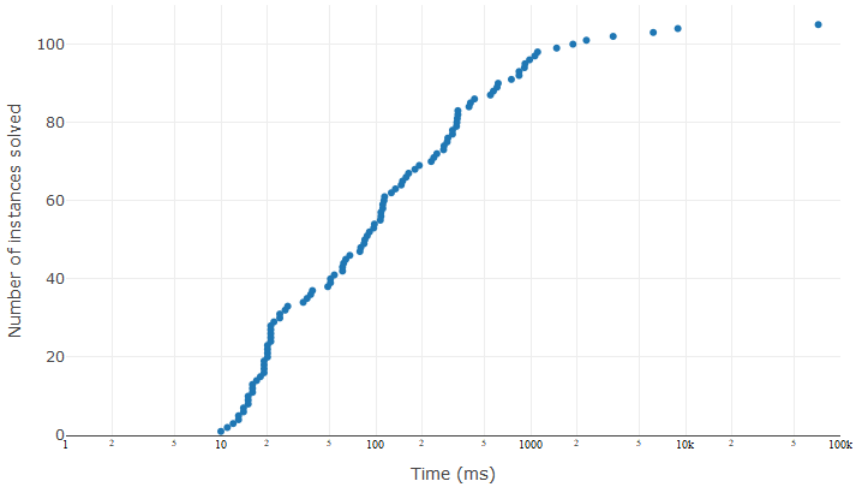


Figure 4.6 Performance profile curve for minimizing RD with the CP algorithm

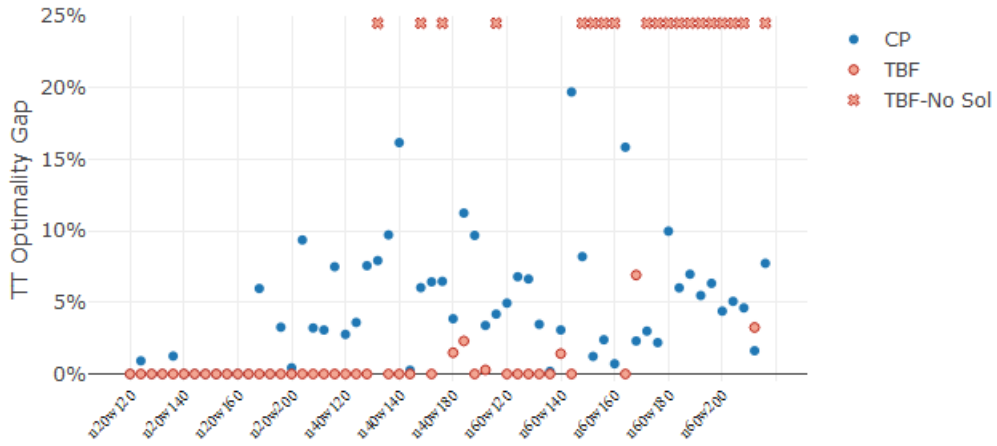


Figure 4.7 Optimality gap when minimizing TT using each algorithm

Table 4.7 Number of instances solved to optimality with respect to a time limit when minimizing TT using each algorithm

	# instances solved	
Time limit (s)	CP	TBF
1	0	6
5	0	11
30	0	31
120	1	34
300	2	35

CHAPITRE 5 ARTICLE 2 : COMPACT ROUTES FOR PARCEL DELIVERY BY POSTAL SERVICES

Cet article a été écrit par A. Bretin, G. Desaulniers et L.-M. Rousseau et soumis à *Computers and Operations Research* en février 2021. De manière analogique au chapitre précédent, les modifications effectuées par rapport au manuscrit soumis à *Computers and Operations Research* apparaissent en rouge. De plus, une section préliminaire (5.0) apporte quelques précisions méthodologiques empruntées aux travaux de Prescott-Gagnon *et al.* [50], sur lesquels nous nous sommes appuyés pour écrire cet article.

5.0 Description préliminaire

Afin de rendre la lecture de la thèse plus compréhensible, nous donnons dans cette section des détails sur l'algorithme introduit par Prescott-Gagnon *et al.* [50] permettant de résoudre des problèmes de tournées de véhicules avec fenêtres de temps. Cette méthode est une recherche à voisinages larges (LNS pour *Large Neighborhood Search* en anglais), c'est à dire une approche itérative dans laquelle la solution courante va être partiellement détruite puis reconstruite à de nombreuses reprises dans le but de trouver des solutions améliorantes. Dans le cas de l'article [50], la reconstruction se fait par une méthode de génération de colonnes heuristique, basée sur de la recherche taboue. Il faut aussi noter que dans cette méthode, deux phases se succèdent. La première, intitulée VNR pour *Vehicle Number Reduction*, a pour but de minimiser le nombre de véhicules utilisés. Nous notons m_{ub} le nombre de véhicules de la meilleure solution réalisable à la fin de cette phase. La seconde, TDR pour *Traveled Distance Reduction*, a pour objectif de minimiser la distance totale parcourue par l'ensemble des véhicules. Le nombre de véhicules n'est pas contraint, mais le coût d'utilisation d'un véhicule supplémentaire étant souvent prohibitif, la meilleure solution à la fin de la phase TDR contient dans la pratique toujours au plus m_{ub} véhicules. L'algorithme général présenté dans l'article sous forme de diagramme de flux (Figure 5.5) est aussi explicité dans le pseudo-code présenté dans l'algorithme 2. Ce dernier permet principalement de comprendre le fonctionnement des différentes phases de recherche. Les détails de ce que nous référons comme une itération algorithmique (étape 25) est détaillé plus en détails dans la section 5.0.1.

Algorithme 2 Fonctionnement général

1: Générer une solution initiale (avec m véhicules)
2: Calculer une borne inférieure sur le nombre de véhicules m_{lb}
3: $m_{ub} \leftarrow m\,;\ i \leftarrow 0\,;\ Phase = VNR$
4: **if** $Phase = VNR$ **then**
5: **if** Il y a des clients non-couverts dans la solution courante **then**
6: **if** $i = I_{VNR}^{max}$ **then**
7: $m_{ub} \leftarrow m_{ub} + 1\,;\ Phase = TDR\,;\ i \leftarrow 0\,;$ Go To 4
8: **else**
9: Go To 25
10: **end if**
11: **else**
12: **if** $m_{ub} = m_{lb}$ **then**
13: $Phase = TDR\,;\ i \leftarrow 0$
14: **else**
15: $i \leftarrow 0\,;\ m_{ub} \leftarrow m_{ub} - 1$
16: **end if**
17: **end if**
18: **else**
19: **if** $i = I_{TDR}^{max}$ **then**
20: STOP
21: **else**
22: Go To 25
23: **end if**
24: **end if**
25: Itération algorithmique
26: $i \leftarrow i + 1$
27: Go To 4

5.0.1 Itération algorithmique

Nous présentons dans cette section une itération complète de l'algorithme, comprenant une phase de destruction ainsi qu'une phase de reconstruction de la solution courante, dans le cas général. Nous faisons abstraction des spéficités propres aux phases de VNR/TDR que nous détaillerons dans la section 5.0.2.

Chaque itération commence par une destruction de la solution courante. Pour ce faire, nous avons recours à divers opérateurs de destructions. Nous nous sommes limités [1] aux opérateurs définis par Prescott-Gagnon *et al.* dans leur article, c'est-à-dire :

1. Nous avons essayé d'implémenter un opérateur de destruction basé sur la compacité. Si les premières occurences de ce dernier dans l'algorithme donnaient parfois de bonnes améliorations, les résultats finaux étaient rarement meilleurs que lorsque nous nous passions de cet opérateur. Devant le manque de résultats convaincants, nous avons préféré passer sous silence cette partie de notre développement dans l'article.

— proximité : détruit des clients relativement à leur proximité spatio-temporelle, prenant en compte notamment les fenêtres de temps éventuelles des clients

— portion de route : détruit des sections de routes complètes pour augmenter la flexibilité de reconstruction

— plus long détour : retire les clients générant de grands détours, qui seraient potentiellement intégrés dans une autre route

— heure de service : retire des clients qui sont visités dans la même plage horaire, encore une fois pour augmenter les chances de reconstructions pertinentes.

Le nombre de clients retirés à chaque phase de destruction Nb_{rem} est paramétrable. Plus celui-ci est grand, plus la reconstruction aura de flexibilité, mais plus celle-ci sera complexe. En effet, une fois la destruction de la solution courante complétée, toutes les portions de routes qui n'ont pas été affectées demeurent gelées et ne pourront donc pas être modifiées pendant cette itération (pas de retrait ni d'ajout de clients possible au coeur de ces sous-chemins gelés). Ne pouvant énumérer toutes les routes réalisables (ensemble noté Ω), la phase de reconstruction prend la forme d'une résolution par génération de colonnes. Elle s'articule autour d'un problème maître restreint (5.1)-(5.4), noté PMR, où l'ensemble des colonnes initiales $\Omega^{dest} \subset \Omega$ correspond au sous-ensemble des routes en mémoire (Ω^{bassin}) qui respectent les sous-chemins gelés imposé par la phase de destruction.

$$\min \quad \sum_{r \in \Omega^{dest}} c_r \alpha_r \tag{5.1}$$

$$\text{subject to :} \quad \sum_{r \in \Omega^{dest}} a_{ir} \alpha_r = 1, \qquad \forall i \in V^C \tag{5.2}$$

$$\sum_{r \in \Omega^{dest}} \alpha_r \leq m_{ub}, \tag{5.3}$$

$$\alpha_r \in \{0, 1\}, \qquad \forall r \in \Omega^{dest}. \tag{5.4}$$

Le sous-problème associé est un plus court chemin élémentaire avec contraintes de ressources que nous résolvons par méthode taboue. La formulation du problème reprend les notations d'un VRPTW traditionnel (voir section 5.2). Cependant, le coût c_{ij} de chaque arc, qui dans notre modélisation est égal au temps de parcours t_{ij} le long de l'arc, est remplacé par son coût réduit

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \eta_i & \text{si } i \in V^C \\ c_{ij} - \mu & \text{si } i = d \end{cases}$$

où η_i et μ sont les variables duales associées aux contraintes (5.2) et (5.3) du PMR. La

génération de colonnes s'effectue alors de la manière décrite par l'algorithme 3.

Entre chaque résolution du PMR (initialement étape (2) puis étapes (9)), nous cherchons à générer des colonnes par méthode taboue. Des colonnes en base sont sélectionnées successivement et aléatoirement jusqu'à ce qu'un critère soit satisfait (boucle 4), généralement un nombre de colonnes déjà générées pendant cette phase de recherche taboue. Les seuls mouvements tabous considérés sont l'ajout et le retrait d'un client dans la route, tout en veillant bien à respecter les sous-chemins gelées en tout temps pour les deux mouvements. L'exploration se fait de manière exhaustive, chaque client admissible étant inséré à chaque position valide de la route, les meilleures solutions à coût réduit négatif étant conservées en mémoire et ajoutées au bassin de colonnes (étape 7). La recherche s'arrête après $NbIterSign^{max}$ consécutives sans amélioration significative (seuil paramétrable). Cependant, si la solution du PMR est fractionnaire (test 14), des stratégies de branchement sont appliquées, principalement basées sur les variables α_r et le processus de recherche continue pour $NbIterSign^{max}$ itérations.

Algorithme 3 Itération de génération de colonnes

1: $NbIterSign \leftarrow 0$
2: Résoudre le PMR avec Ω^{dest}
3: **while** $NbIterSign < NbIterSign^{max}$ **do**
4: **while** Condition d'arrêt tabou non satisfaite **do**
5: Choisir une route r en base
6: Chercher, par méthode taboue, dans le voisinage de r, un ensemble R' de colonnes r' à coûts réduit négatif
7: $\Omega^{dest} = \Omega^{dest} \cup R'$
8: **end while**
9: Résoudre PMR
10: **if** Amélioration significative **then**
11: $NbIterSign \leftarrow 0$
12: **else**
13: $NbIterSign \leftarrow NbIterSign + 1$
14: **if** $NbIterSign = NbIterSign^{max}$ & Solution du PMR fractionnaire **then**
15: Branchement sur le PMR
16: $NbIterSign \leftarrow 0$
17: **end if**
18: **end if**
19: **end while**

5.0.2 Spécifités de la phase VNR

En dehors de certains paramètres sur les nombres d'itérations ou de seuils d'améliorations qui peuvent varier d'une phase à l'autre à des fins de performance, la différence principale

entre VNR et TDR réside dans la gestion des clients nous couverts. Si ceux-ci sont proscrits dans la phase TDR, afin que toutes les solutions demeurent réalisable, dans la phase de VNR où l'on cherche à réduire le nombre de véhicules utilisées, les clients non-couverts sont simplement fortement pénalisés. Pour aider à retrouver une couverture complète des clients, les opérateurs de destruction sont aussi adaptés afin d'aider à leur réinsertion.

5.1 Introduction

This paper deals with parcel delivery performed by postal services, where parcels are normally handled separately from mail and delivered by trucks. Even if parcel delivery by drones is getting increased attention in the literature [see, e.g., 36, 37], we focus on traditional truck deliveries because drones are not yet broadly deployed for this task. The problem that we consider belongs to the class of the vehicle routing problem (VRP). Introduced by [7], the VRP is defined as computing the routes of a fleet of identical capacitated vehicles, originally located at a central depot, such that each customer of a given set is visited exactly once by a vehicle to satisfy its demand, the total load delivered along a route does not exceed vehicle capacity, and a cost function often related to the total distance traveled is minimized. One of its most common variant is the VRP with time windows (VRPTW) which restricts the service at a customer to start within a predefined time interval. For parcel delivery, not all deliveries are subject to time window restrictions because they are present mainly for commercial customers or to respect fast-delivery contracts (*e.g.*, within 24 or 48 hours).

In the scientific literature, many solution algorithms have been proposed to solve the VRP and its variants. For many of them, the leading exact algorithms are branch-price-and-cut algorithms as surveyed by [38]. Heuristics are, however, unavoidable to solve large instances. Some of the most performing ones are the hybrid genetic algorithm of [56] and the memetic algorithm of [57]. For the VRPTW, the branch-and-price-based large neighborhood search (LNS) algorithm of [50] also produced satisfying computational results [see 79]. Our work is based on this matheuristic.

In most parcel delivery companies, especially postal agencies, the objective of the parcel delivery problem considered is to find a solution that offers a good compromise between the usual routing cost, which is proportional to the distance traveled (or, sometimes, the time spent traveling), and the compactness of the routes which can be measured as discussed below. The compactness criterion is motivated by two main factors: territory knowledge and route acceptability. Indeed, when routes are compact, it is easier to assign to each driver a route that is restricted to a territory (neighborhood) that is well known by the driver, thus, increasing his/her effectiveness and reducing the driver's stress. Also, as indicated to us by our industrial partner GIRO (which develops route optimization software for postal services), compact routes are better accepted by the route planners and the drivers for the following reasons. On the one hand, compact routes remain more or less in the same neighborhood and can, thus, be assigned more easily by the planners to the drivers according to the drivers' knowledge of specific neighborhoods. On the other hand, compact solutions tend to reduce route overlapping, lowering the chances that several drivers are in the same area at the same

time, which has been reported as frustrating for the drivers who believe that intersecting routes are inefficient. Finally, compact routes are also appreciated by the drivers of some companies that reward them for each parcel delivered in hand because compactness favors short returns to absent customers.

This paper focuses on the concept of route compactness which is not well referenced in the literature (see Section 5.2.1). We propose to measure the compactness of a route as the area of the convex hull of the locations of its visited customers (thus, excluding the depot). For the sake of conciseness, this convex hull is called the convex hull of the route. Because computing the area of a polygonal convex hull may be too time-consuming to be performed within an optimization algorithm, we introduce two approaches to approximate the convex hull of a route. The first one, purely geographical, is a rectangular approximation. The second one, which is more easily adaptable to handle road network specificities, is a disk approximation where the disk diameter may be defined not only as the Euclidean distance between two customer locations but also, for example, as the travel time between two locations.

To assess the tradeoff between routing costs and route compactness, we adapt the branch-and-price-based LNS algorithm of [50] such that it can handle route compactness penalties as well as other features related to route duration as discussed below. Note that our intent is not to devise the most efficient algorithm for solving the problem at hand, but rather to have an algorithm that is applicable to compare the proposed approximate compactness measures. Our computational experiments, performed on academic and on instances derived from a real-world dataset, allow such a comparison and to evaluate the additional computational burden for dealing with route compactness in this LNS heuristic.

Our contribution is thus threefold. First, we introduce a new variant of the VRPTW that we call the VRPTW with route compactness and denote Comp-VRPTW. Second, we propose and study different route compactness measures. Third, we modify an existing heuristic so that it can handle these route compactness measures.

The rest of this paper is structured as follows. In Section 5.2, we define the parcel delivery problem considered *i.e.*, the Comp-VRPTW, introduce the proposed route compactness measures, and provide a mathematical formulation for the Comp-VRPTW. In Section 5.3, we describe the enhanced branch-and-price-based LNS algorithm. Computational results on academic and industry-derived instances are reported in Sections 5.4 and 5.5, respectively. Finally, a short conclusion is drawn in Section 5.6.

5.2 Problem definition and mathematical formulation

In postal services, the amount of customers to visit in a route is usually quite large (between 50 and 100), whereas the weight or volume of every single delivery is rather limited. For this reason, the routes are typically time-constrained but not necessarily capacity-constrained. This time constraint is referred to as the route duration (RD) constraint. Note that route duration is closely related to the distance traveled, but also to the set of customers visited, because the service time may vary from one customer to another. For a given route, it is usually computed as the sum of the total travel time, service time, and waiting time if any.

The Comp-VRPTW can be stated as follows. Given a set of customers V^C to service exactly once and a homogeneous fleet of capacitated vehicles L, the problem consists in designing compact feasible routes such that a weighted sum of the routing costs and the compactness costs is minimized. In the following, we detail route feasibility and the cost structure.

Each customer $i \in V^C$ is located at coordinates (X_i, Y_i). The service at some customers $i \in V^C$ must start within a time window $[\underline{w}_i, \bar{w}_i]$. For the other customers, we assume that they are associated with an unconstraining time window $[\underline{w}_i, \bar{w}_i]$, where $\underline{w}_i = 0$ and $\bar{w}_i$ is equal to a large value W. Note that a vehicle can arrive earlier than $\underline{w}_i$ at customer i but must wait until $\underline{w}_i$ before starting service. A service time s_i and a demand q_i are associated with each customer $i \in V^C$. All routes start and end at the same depot which is denoted d. The departure time of a route is not fixed but must be greater than or equal to 0. Let $V = V^C \cup \{d\}$ be the set of all customer and depot locations. With each pair of distinct locations $(i, j) \in V^2$, $i \neq j$, are associated a travel time t_{ij} as well as a travel cost c_{ij} which is generally proportional to t_{ij}.

A feasible route r is given by a sequence of locations $(v_0, v_1, \ldots, v_p)$ with $p \geq 2$ such that v_0 and v_p represent the depot at the beginning and the end of the route, respectively, $v_i \in V^C$, $i \in \{1, \ldots, p-1\}$, and $v_i \neq v_j$ for all $i, j \in \{1, \ldots, p-1\}$, $i \neq j$. It is associated with a schedule $(T_{v_0}, T_{v_1}, \ldots, T_{v_p})$, where T_{v_0} and T_{v_p} denote the route starting and ending times, respectively, and T_{v_i} the start of service time at customer v_i, $i \in \{1, \ldots, p-1\}$. This schedule must respect the customer time windows, *i.e.,* it must satisfy the following conditions:

$$T_{v_0} \geq 0 \tag{5.5}$$

$$T_{v_{i-1}} + s_{v_{i-1}} + t_{v_{i-1},v_i} \leq T_{v_i}, \qquad \forall i \in \{1, \ldots, p-1\} \tag{5.6}$$

$$T_{v_i} \in [\underline{w}_i, \bar{w}_i], \qquad \forall i \in \{1, \ldots, p-1\}, \tag{5.7}$$

with $s_{v_0} = 0$. When the times $(T_{v_0}, T_{v_1}, \ldots, T_{v_p})$ are seen as variables, the (minimal) duration

U_r of route r is equal to the optimal value of the linear program:

$$\begin{aligned} U_r \quad = \quad & \min \quad T_{v_p} - T_{v_0} \\ & \text{subject to:} \quad (5.5)\text{–}(5.7). \end{aligned} \tag{5.8}$$

To avoid excessive driver fatigue, this duration must not exceed U^{max}, a user-defined parameter which generally depends on the country's labor legislation and the drivers' union rules. Finally, when the capacity of a vehicle Q^{max} must be taken into account, the total demand of the visited customers along route r must not exceed this capacity, *i.e.,* $\sum_{i=1}^{p-1} q_i \leq Q^{max}$.

The cost c_r of route r is given by

$$c_r = \tau_r + \lambda \kappa_r, \tag{5.9}$$

where τ_r and κ_r are the routing cost and the compactness measure of route r, respectively, and $\lambda \geq 0$ is a weight that can be adjusted to put more or less emphasis on the compactness criterion. The routing cost τ_r is given by $\tau_r = F + \sum_{i=0}^{p-1} c_{v_i,v_{i+1}}$, where F is a fixed cost sufficiently large to ensure that the number of vehicles used is minimized.

In the following, we discuss different compactness measures that can be used for κ_r (Section 5.2.1) and provide a mathematical formulation for the Comp-VRPTW (Section 5.2.2).

5.2.1 Compactness measures

The scientific literature on vehicle route compactness is still scarce. For waste collection, [30] introduced a shape metric to measure the compactness of a route. This measure is computed, in practice, as the sum of the Euclidian distances between the locations visited in the route and its gravity center. It has the disadvantage of considering individually each visited location which may result in an overestimated measure when several locations are close together but relatively far from the gravity center. In addition, to design better solutions, these authors consider how the route convex hulls overlap in the heuristic they propose. Trying to identify which characteristics favor good VRP solutions, [32] defined two new measures of compactness. The first sums over all visited customers in a route the Euclidian distances between a customer and the straight line passing through the depot and the gravity center of the route. The second rather sums the angles between this line and the line linking the depot and a customer. These measures tend to generate petal-shaped routes around the depot, which is not necessarily the most desired pattern in postal services because the neighborhoods nearby the depot are covered by too many drivers. Another definition of compactness was proposed by [31] for a sales territory design problem which is defined on a graph. The compactness of a territory composed of a subset of nodes in this graph is

computed as the maximum distance between each node in this subset and a center that corresponds to one of the subset nodes. As in [30], the choice of the center is a decision variable but it is limited to a discrete set of possibilities.

To take into account a route compactness measure while solving the Comp-VRPTW, this measure must meet the following two criteria:

1. It should be based on the customer set only and not be sequence-dependent;
2. It should be relatively fast to compute or at least to update.

Criterion 1 captures the idea of minimizing some geometric area in which the driver has to work, without penalizing any detour inside this area that may be caused, for instance, by a customer with a tight time window. In other words, a customer located inside the convex hull of a route should belong to this route. Furthermore, according to our industrial partner, the depot location should not be taken into consideration when measuring the compactness of a route. Criterion 2 stems from the application context. Parcel delivery in postal services is very dynamic, and routes from one day to another may be very different and should be recomputed every night in a relatively narrow time slot. For this reason, the computational impact of considering compactness should be limited.

Measuring the compactness of a route as the area of the convex hull of its visited customers satisfies the first criterion, but not the second. Therefore, after presenting this time-consuming measure first, we introduce three approximate measures (the last one combining the other two) which are less time-consuming.

5.2.1.1 Exact computation

In this section, we briefly explain how the area of the convex hull of a route can be computed exactly. For our computational experiments, we performed this computation *a posteriori* on the final solution to serve as a comparison for evaluating the proposed approximate measures.

To compute this area, the convex hull has to be identified first. The most efficient ways to proceed are with the algorithm of [80] in $\mathcal{O}(n \log n)$ or that of Chan [81] in $\mathcal{O}(n \log h)$, where n is the number of customers in the route and h the number of these customers defining the convex hull. Because h is unknown in practice, Chan's algorithm starts by choosing a parameter m and is successful if and only if $m \geq h$ (the complexity is then $\mathcal{O}(n \log m)$). In the worst case, $m = n$ and we assume in the following that the worst-case complexity of identifying the convex hull of a route is given by $\mathcal{O}(n \log n)$.

Once the convex hull is identified, the computation of its area A is quite straightforward. Let N be the number of vertices of the corresponding polygon and assume that these vertices

(X_1, Y_1) to (X_N, Y_N) are ordered, following the boundary of the polygon, either clockwise or counterclockwise. The area is then given by:

$$A = \frac{1}{2} \left| \sum_{i=1}^{N} X_i Y_{i+1} - X_{i+1} Y_i \right|,$$

where $(X_{N+1}, Y_{N+1}) = (X_1, Y_1)$. Thus, this computation requires $\mathcal{O}(n)$ operations.

In the following, we denote by κ_r^E the area A computed for a route r and call it the exact compactness measure of route r.

5.2.1.2 Rectangle approximation

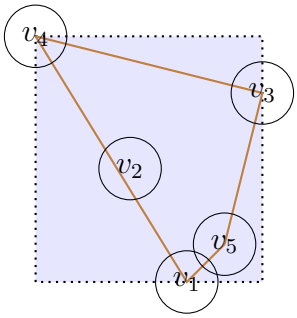
Our first suggestion is to approximate the convex hull of a route $r = (v_0, v_1, \ldots, v_p)$ with a rectangle that contains all the visited customers in r and whose sides are parallel to the axes of the geographical coordinates. Let

$$x_r^{Min} = \min_{i \in \{1,\ldots,p-1\}} X_{v_i}, \qquad x_r^{Max} = \max_{i \in \{1,\ldots,p-1\}} X_{v_i},$$
$$y_r^{Min} = \min_{i \in \{1,\ldots,p-1\}} Y_{v_i}, \qquad y_r^{Max} = \max_{i \in \{1,\ldots,p-1\}} Y_{v_i},$$

be the minimum and maximum values of the X- and Y-coordinates of the visited customers. The area A of the rectangles defined by the points (x_r^{Min}, y_r^{Min}), (x_r^{Min}, y_r^{Max}), (x_r^{Max}, y_r^{Min}), and (x_r^{Max}, y_r^{Max}) is given by $A = (x_r^{Max} - x_r^{Min})(y_r^{Max} - y_r^{Min})$. However, to avoid the critical case where all customers are colinear on a line parallel to an axis, we introduce a minimum width and length μ (*e.g.*, 1% of the largest dimension) and propose the following rectangle compactness measure, denoted κ_r^R:

$$\kappa_r^R = \min\{\mu, x_r^{Max} - x_r^{Min}\} \times \min\{\mu, y_r^{Max} - y_r^{Min}\}.$$

In Figure 5.1, we present an example with five customers for which $x_r^{Min} = -8$, $x_r^{Max} = 4$, $y_r^{Min} = 0$, and $y_r^{Max} = 13$, yielding $\kappa_r^R = 12 \times 13 = 156$. Graphically, one can see that the rectangle is a relatively poor approximation of the route convex hull (in brown) because there is a significantly large empty space in the lower-left corner of the rectangle. Allowing rotated rectangles could potentially improve the approximation. However, for computational efficiency, we consider only a subset of the possible rotation angles. Furthermore, instead of rotating the rectangles throughout the execution of the algorithm, we precompute the rotated coordinates of each customer once at the beginning of the algorithm for each considered angle, allowing a fast computation of the area of the rectangles using only the minimum and



Points	(X_i, Y_i)
v_1	(0,0)
v_2	(-3,6)
v_3	(4,10)
v_4	(-8,13)
v_5	(2,2)

Figure 5.1 A rectangle approximating the convex hull of a route

maximum functions as described above. Note that rotating all the points does not change their relative location one to another, so the area remains consistent. For a rotation angle θ, the computation of the new coordinates $(X^{\theta}_{v_i}, Y^{\theta}_{v_i})$ for a customer v_i is performed using the following formula:

$$\begin{pmatrix} X^{\theta}_{v_i} \\ Y^{\theta}_{v_i} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} X_{v_i} \\ Y_{v_i} \end{pmatrix}$$

where $(X_{v_i}, Y_{v_i}) = (X^0_{v_i}, Y^0_{v_i})$ are the original coordinates of the point.

Let Θ be the subset of rotation angles considered, *e.g.*, $\Theta = \{0, 15, 30, 45, 60, 75\}$ (in degrees). Because of the symmetry between the rectangles, one can easily observe that it is sufficient to select only angles in the semi-open interval $[0, 90[$. Then, for each angle $\theta \in \Theta$ and corresponding dataset of coordinates $(X^{\theta}_{v_i}, Y^{\theta}_{v_i})$, $i \in \{1, \ldots, p-1\}$, we compute as above the minimum and maximum values of the X^{θ}- and Y^{θ}-coordinates:

$$x_r^{Min}(\theta) = \min_{i \in \{1,\ldots,p-1\}} X^{\theta}_{v_i}, \qquad x_r^{Max}(\theta) = \max_{i \in \{1,\ldots,p-1\}} X^{\theta}_{v_i},$$
$$y_r^{Min}(\theta) = \min_{i \in \{1,\ldots,p-1\}} Y^{\theta}_{v_i}, \qquad y_r^{Max}(\theta) = \max_{i \in \{1,\ldots,p-1\}} Y^{\theta}_{v_i},$$

which are used to compute a rectangle compactness measure associated with θ:

$$\kappa_r^R(\theta) = \min\{\mu, x_r^{Max}(\theta) - x_r^{Min}(\theta)\} \times \min\{\mu, y_r^{Max}(\theta) - y_r^{Min}(\theta)\}.$$

Finally, to determine the final rectangle compactness measure of route r. we take the minimum of the measures computed over all considered angles, *i.e.*,

$$\kappa_r^R = \min_{\theta \in \Theta} \kappa_r^R(\theta).$$

Let us return to the example in Figure 5.1 which provided $\kappa_r^R(0) = 156$. If we choose $\Theta = \{0, 30, 60\}$, we also have to consider the rectangles drawn in Figures 5.2 and 5.3 which yield $\kappa_r^R(30) = 14.16 \times 10.66 = 150.95$ and $\kappa_r^R(60) = 15.26 \times 8.89 = 135.66$. Finally, we obtain $\kappa_r^R = \min\{156, 150.95, 135.66\} = 135.66$, a significant improvement over the no-rotation value $\kappa_r^R(0) = 156$. As it will be discussed in Section 5.4.2, the more angles in set Θ, the better the approximation. However, considering more angles increases computational times and, therefore, a compromise between measure precision and computational time must be achieved.

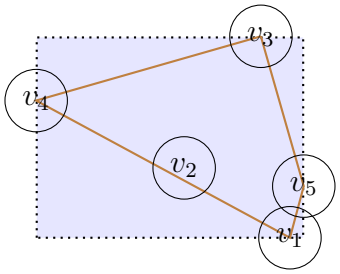
5.2.1.3 Disk approximation

As a second approximation of the convex hull of a route $r = (v_0, v_1, \ldots, v_p)$, we propose to use a disk whose diameter D_r is given by the maximum Euclidian distance between two customers visited in r and whose boundary passes by two customers yielding this maximum distance. More precisely, let $V_r = \{v_1, \ldots, v_{p-1}\}$ be the set of visited customers in r and e_{v_i,v_j} the Euclidian distance between customers v_i and v_j in V_r. Then, $D_r = \max_{(v_i,v_j)\in V_r^2} e_{v_i,v_j}$. The corresponding disk compactness measure, denoted κ_r^D, is then given by the disk area:

$$\kappa_r^D = \left(\frac{D_r}{2}\right)^2 \pi.$$

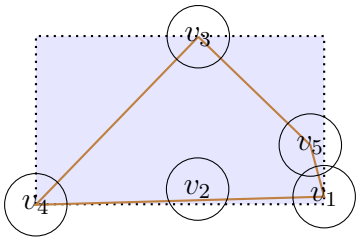
For the example in Figure 5.1, the most distant customers are v_1 and v_4, yielding a diameter of $\sqrt{233}$ and a disk compactness measure of $\kappa_r^D = (\sqrt{233}/2)^2\,\pi = 183.0$. A graphical representation of the corresponding disk is given in Figure 5.4, which shows that the disk does not necessarily contain all visited customers and, therefore, the whole route convex hull. Nevertheless, it is easy to see that, when visiting the same number of customers, routes with smaller disk areas are, in general, more compact than those with larger areas.

One can notice that minimizing the area of a single disk is equivalent to minimizing the diameter of this disk. However, this equivalence does not hold when seeking to minimize the sum of several disk areas (induced by multiple routes like in the Comp-VRPTW). Indeed, consider an example with two disks and two distinct solutions with an equal sum of the diameters: diameters 2 and 8 for the first solution and diameters 4 and 6 for the second. Computing the sums of the corresponding disk areas yields unequal values, namely, $\pi + 16\pi = 17\pi$ and $4\pi + 9\pi = 13\pi$. Nevertheless, using the diameter directly as a compactness measure, *i.e.*, setting $\kappa_r = D_r$, may be suitable when interesting alternatives to the Euclidean distance make sense. In fact, in practice, when travel times are calculated according to a certain road network, including some traffic regulations (one-way streets, prohibited left turns, ...)



Points	(X_i^{30}, Y_i^{30})
v_1	(0,0)
v_2	(-5.60,3.70)
v_3	(-1.54,10.66)
v_4	(-13.43,7.26)
v_5	(0.73,2.73)

Figure 5.2 Rotated rectangle approximation ($\theta = 30°$)



Points	(X_i^{60}, Y_i^{60})
v_1	(0,0)
v_2	(-6.70,0.40)
v_3	(-6.66,8.46)
v_4	(-15.26,-0.43)
v_5	(-0.73,2.73)

Figure 5.3 Rotated rectangle approximation ($\theta = 60°$)

and geographical constraints (rivers with few bridges, mountains with or without tunnels, ...), computing the route compactness with real network distances (shortest path lengths) or travel times may be more appropriate than with distances as crow flies. However, when doing so, any geographical area comparison becomes meaningless.

5.2.1.4 Hybrid route compactness measure

The quality of the two approximations introduced above depends on the shape of the route. As the shape of the routes considered in the proposed algorithm may vary significantly, mixing both rectangle disk compactness measures should lessen and rectify low-quality fits. Therefore, we define κ_r^H, the hybrid compactness measure of a route r as a convex combination of κ_r^R and κ_r^D, *i.e.*,:

$$\kappa_r^H = \omega^R \kappa_r^R + \omega^D \kappa_r^D,$$

where $\omega^R, \omega^D \geq 0$ and $\omega^R + \omega^D = 1$. These parameters are user-defined weights that can be adjusted to favor one measure over the other. Note that this hybrid measure encompasses the two previous measures. Indeed, setting $\omega^R = 1$ and $\omega^D = 0$ yields $\kappa_r^H = \kappa_r^R$, whereas setting $\omega^R = 0$ and $\omega^D = 1$ yields $\kappa_r^H = \kappa_r^D$.

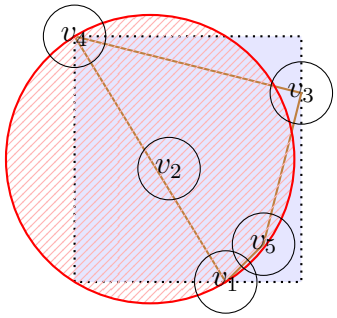


Figure 5.4 Disk and rectangle approximations

5.2.2 Mathematical formulation of the Comp-VRPTW

Let Ω be the set of all feasible routes. According to (5.9), the cost of a route $r \in \Omega$ is expressed as $c_r = \tau_r + \lambda\kappa_r$. In theory, we would like to set $\kappa_r = \kappa_r^E$. However, given the computational complexity of considering κ_r^E during the solution process, we rather use $\kappa_r = \kappa_r^H$ to yield $c_r = \tau_r + \lambda^R\kappa_r^R + \lambda^D\kappa_r^D$, where $\lambda^R = \lambda\omega^R$ and $\lambda^D = \lambda\omega^D$. Let a_{ir}, $i \in V^C$, $r \in \Omega$, be a binary parameter equal to 1 if route r covers customer i and to 0 otherwise. Finally, with each route $r \in \Omega$, we associate a binary variable α_r that takes value 1 if route r belongs to the solution and to 0 otherwise.

With this notation, the Comp-VRPTW that we consider can be formulated as follows:

$$\min \quad \sum_{r\in\Omega} c_r\alpha_r \tag{5.10}$$

$$\text{subject to:} \quad \sum_{r\in\Omega} a_{ir}\alpha_r = 1, \qquad \forall i \in V^C \tag{5.11}$$

$$\sum_{r\in\Omega} \alpha_r \leq m_{ub}, \tag{5.12}$$

$$\alpha_r \in \{0,1\}, \qquad \forall r \in \Omega. \tag{5.13}$$

Objective function (5.10) aims at minimizing a weighted sum of the total routing and compactness costs. Constraints (5.11) ensure customer coverage, while constraint (5.12) restricts the number of routes used in the solution to $m_{ub} = |L|$, the maximum number of vehicles available in fleet L. Binary requirements on the decision variables are imposed through constraints (5.13).

5.3 Solution algorithm

To solve the Comp-VRPTW, we propose to adapt the branch-and-price-based LNS heuristic developed by [50] for tackling the VRPTW. Starting from an initial solution, this algorithm proceeds in two phases: the first one aims for a vehicle number reduction (VNR) which is aligned with the primary objective of the Comp-VRPTW (minimizing the number of vehicles assuming that the vehicle fixed cost F is sufficiently large), whereas the second seeks to achieve a traveled distance reduction (TDR). In the VNR phase, the upper bound m_{ub} defining the right-hand side of constraint (5.12) is first set to the number of vehicles used in the initial solution minus one. If a feasible solution can be found within a certain number of LNS iterations I_{max}^{VNR}, m_{ub} is decremented again by one and new LNS iterations are performed. This process continues until no feasible solution can be found for a given bound or m_{ub} reaches a pre-computed lower bound m_{lb}. In the former case, m_{ub} is increased by one before moving on to the TDR phase. Note that with this strategy, there is no need to consider the fixed cost F in both phases. Furthermore, given the additional computational burden of handling the compactness costs, they are not taken into account in the VNR phase.

The solution process applied in both phases is roughly the same, but differs by the parameter setting. The general behavior of the LNS algorithm is presented in Figure 5.5. The main loop of the algorithm consists of the following sequence of steps. Starting from the current solution, we partially destroy it using a destruction operator, which is chosen randomly from a predefined set of four operators using a roulette-wheel procedure similar to that of [51]. This procedure favors operators that have been successful in the recent iterations, *i.e.*, their application yielded an improved solution at the end of the LNS iteration. The chosen destruction operator removes customers from the routes in the current solution, leaving fixed partial routes.

After destruction, a branch-and-price heuristic applied to model (5.10)–(5.13) is called to rebuild a solution taking into account the fixed partial routes. The branch-and-price heuristic applies heuristic column generation to solve linear relaxations and a diving branching strategy to derive an integer solution. Column generation is an iterative algorithm that solves at each iteration a restricted master problem, i.e., the linear relaxation of (5.10)–(5.13) restricted to a subset of its variables (the so-called columns and representing already generated routes), and a subproblem which finds negative reduced cost routes to add to the restricted master problem. Usually, column generation stops when there are no more negative reduced cost routes. In our case and as proposed by [50], we solve the subproblem heuristically using the tabu search heuristic of [42], possibly failing to find a negative reduced cost route when some exist. This heuristic offers a lot of flexibility (in particular to handle the compactness

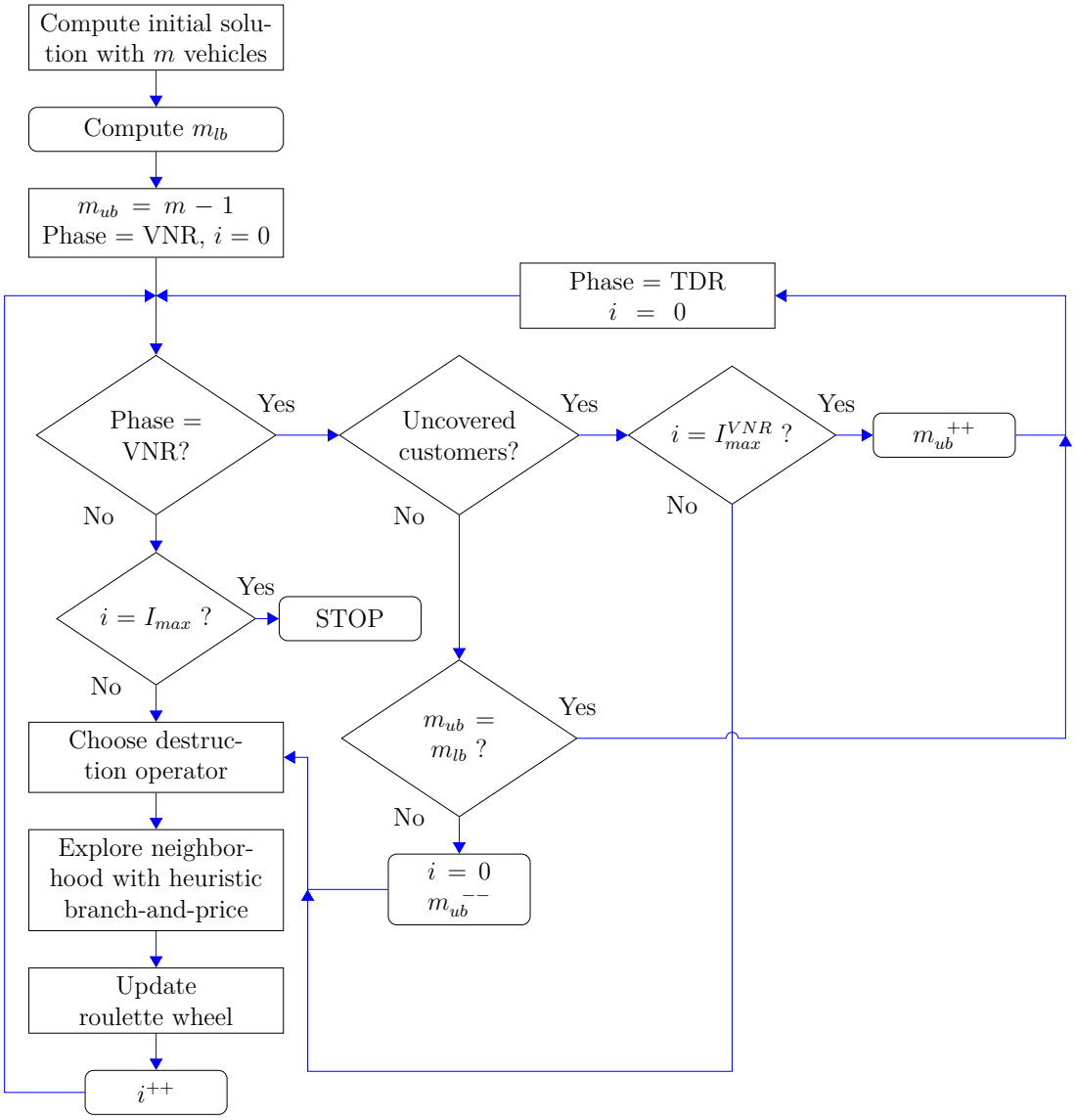


Figure 5.5 Algorithm flowchart

costs) and, typically, yields shorter computational times. In fact, at each column generation iteration, it is applied in a multi-start fashion, starting each time from a column in the current basis of the restricted master problem, for a very limited number of tabu search iterations. Only two types of moves are considered, namely, customer insertion and customer removal. During the TDR phase where the compactness costs are taken into account, the evaluation of the possible insertion and removal moves must consider their impact on the compactness measure of the routes.

The branch-and-price heuristic generally yields a new solution that becomes the current solution even if it does not improve over the previous one. It may also happen that this

heuristic fails to find any solution, in which the previous solution is kept as the current one. The algorithm moves on to update the roulette-wheel statistics before starting a new iteration. The other steps in Figure 5.5 are invoked to initialize the LNS heuristic and to determine when to lower the m_{ub} bound, when to switch from the VNR to the TDR phase, and when to stop the LNS heuristic. Further details about this branch-and-price-based LNS heuristic can be found in [50].

Below, we focus on the description of the algorithm adaptations that were necessary to tackle the Comp-VRPTW. More precisely, we discuss how we compute the initial solution and how the tabu search column generator handles the route duration constraint and the compactness costs.

5.3.1 Initial solution

Finding a good initial solution helps to reduce the computational time of the VNR phase, but it should not be too much time consuming. To favor compactness minimization in the TDR phase, we generate this solution by first splitting the set of customers into disjoint subsets according to geographical boundaries such that each subset contains approximately the same number of customers. After ordering customers according to their X_i values, the instance is splitted in A sub-parts of equal size. Then, customers of each sub-part are ordered according to their Y_i values and the sub-part is then divided into B new sub-parts. The numbers of subsets chosen ($A \times B$) depends on the instance size. A graphical example is presented in Figure 5.6 for an instance with N customers that have to be partitioned into four subsets (if $N \bmod 4 \neq 0$ the number of customers in a subset is equal to $\lfloor N/4 \rfloor$ or $\lceil N/4 \rceil$). Each customer subset yields a sub-instance which is solved by a simple VRPTW constraint programming algorithm, restricted to a tight time limit (*e.g.*, 10 seconds). The constraint programming model used is described in Appendix 5.A. As the sub-instances are completely independent, the computation of the initial solution can be performed in parallel (this is not the case for our tests). The impact of the quality of the initial solution is discussed in Section 5.4.4.

5.3.2 Adapted tabu search route generator

As mentioned above, some adaptations to the tabu search column (route) generator are required to be able to solve the Comp-VRPTW with the branch-and-price-based LNS framework of [50]. They aim at computing the duration of a route and measuring its compactness. We recall that only two move types are applied in this tabu search heuristic, namely, customer insertion and customer removal.

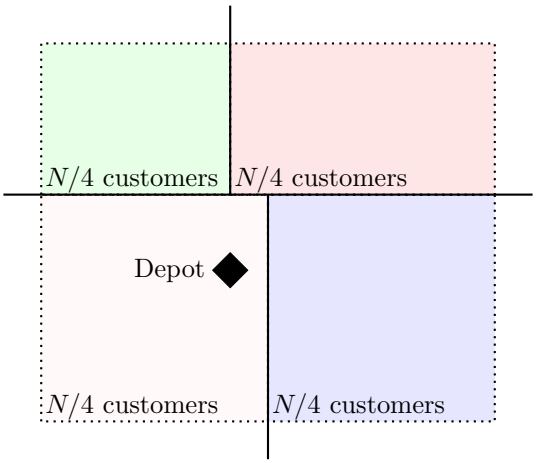


Figure 5.6 Example of an instance splitting for finding an initial solution

5.3.2.1 Route duration computation

In the Comp-VRPTW, the routes are subject to a maximum duration constraint, *i.e.*, the duration U_r of each route $r \in \Omega$ must be such that $U_r \leq U^{max}$. In the tabu search algorithm, there is no need to verify this constraint when removing a customer from a feasible route r. On the other hand, a feasibility check must be performed when inserting a customer in r to yield a new route r' because $U_{r'} \geq U_r$. Recall that the duration U_r of a route r can be computed as the optimal value of the linear program (5.5)–(5.8). However, solving a linear program might be prohibitively time-consuming in a tabu search algorithm. We rather use a simple procedure with complexity $\mathcal{O}(|V_r|)$, where V_r is the subset of customers visited in route r. To understand the idea behind this procedure, we first discuss the example of a route $r = (v_0, v_1, \ldots, v_4, v_5)$ with four customers in Table 5.1. We assume that $s_{v_i} = 2$ for all $i = 1, \ldots, 4$, and $t_{v_{i-1},v_i} = 3$ for all $i = 1, \ldots, 5$. The first row in this table displays the time window associated with each location where $W = 50$. The second row presents a feasible schedule $(T_{v_0}, T_{v_1}, \ldots, T_{v_5})$ for route r where the service at each customer starts as early as possible and the departure and arrival times at the depot are also as early as possible. With this schedule, the duration of route r is equal to $T_{v_5} - T_{v_0} = 36$. Finally, the last row provides a schedule $(T^*_{v_0}, T^*_{v_1}, \ldots, T^*_{v_5})$ that minimizes route duration, yielding $U_r = T^*_{v_5} - T^*_{v_0} = 29$. To obtain this schedule, the departure time from the depot was shifted forward by 7 compared to the first schedule, where 7 is equal to the slack between $\bar{w}_{v_1}$ and T_{v_1}. This slack was compressed to save waiting time before servicing customers v_2 and v_4 and to reduce the duration of the route. Note that the times remain the same for the last two locations v_4 and v_5 because it is not possible to further move ahead the departure time from the depot without violating the time window of a customer before v_4.

Table 5.1 Two schedules for the same route with four customers

	v_0	v_1	v_2	v_3	v_4	v_5
$[\underline{w}_{v_i}, \bar{w}_{v_i}]$	$[0, 50]$	$[3, 10]$	$[12, 25]$	$[0, 50]$	$[31, 35]$	$[0, 50]$
T_{v_i}	0	3	12	17	31	36
$T^*_{v_i}$	7	10	15	20	31	36

This example illustrates for a simple case that the minimal duration of a route can be computed by assigning first the earliest service start times to the customers, while keeping track of the slack that can be used to shift forward the time of departure from the depot to avoid waiting at a customer. This is the rationale behind the procedure that we use to compute the duration of a route. The pseudo-code of this procedure is given in Algorithm 4, where S stores the current slack time available. For each customer v_i, it first computes its earliest arrival time in Step 5. If this arrival occurs before the opening of the customer time window (Step 6), we try to shift forward the departure time from the depot (Step 7) to avoid all waiting time if possible or by the amount of slack time available otherwise. Note that the times associated with the customers $v_1, v_2, \ldots, v_{i-1}$ are also shifted but there is no need to make those computations if we only search for the route duration. Next, the slack time is reduced by the amount consumed (Step 8) and the service start time at v_i is set to $\underline{w}_{v_i}$ (Step 9). Finally, the slack time is updated with respect to the time window upper bound $\bar{w}_{v_i}$ in Step 11 before moving on to the next visited location in route r.

This procedure is not as fast as that of [56] which exhibits an $\mathcal{O}(1)$ complexity. Nevertheless, it seems sufficient for our tests because, in postal services, the majority of the customers do not have a time window and, therefore, waiting before servicing a customer is rare. In this case, the duration U_r of a route $r = (v_0, \ldots, v_p)$ is often well approximated by $T_{v_p} - T_{v_1} + t_{v_0,v_1}$, where T_{v_1} and T_{v_p} are the earliest service start time at v_1 and arrival time at v_p, two variables that are available in the tabu search algorithm when dealing with time windows. In fact, $U_r \leq T_{v_p} - T_{v_1} + t_{v_0,v_1}$ and, therefore, if $T_{v_p} - T_{v_1} + t_{v_0,v_1} \leq U^{max}$, there is no need to compute U_r. Finally, because the best position according to the travel time to insert a customer in a route is often the position that yields the minimal route duration when waiting is not frequent, the route duration constraint is only checked once for each customer that can be inserted in a route, namely, after finding this best insertion position.

5.3.2.2 Compactness cost update

Unlike the route duration which is subject to a hard constraint, route compactness is seen as a preference that incurs a cost. Therefore, this cost as to be updated, if needed, at

Algorithm 4 Route duration computation

Input: Route $r = (v_0, v_1, \ldots, v_p)$
$T_{v_0} \leftarrow 0$
$S \leftarrow W$
for $i = 1, \ldots, p$ **do**
    $T_{v_i} \leftarrow T_{v_{i-1}} + s_{v_{i-1}} + t_{v_{i-1},v_i}$
    if $T_{v_i} < \underline{w}_{v_i}$ **then**
        $T_{v_0} \leftarrow T_{v_i} + \min\{S, \underline{w}_{v_i} - T_{v_i}\}$
        $S \leftarrow S - \min\{S, \underline{w}_{v_i} - T_{v_i}\}$
        $T_{v_i} \leftarrow \underline{w}_{v_i}$
    end if
    $S \leftarrow \min\{S, \bar{w}_{v_i} - T_{v_i}\}$
end for
Return $T_{v_p} - T_{v_0}$

each customer insertion and removal. Note, however, that this computation does not have to be performed for every possible insertion position of a customer because the proposed compactness measures do not depend on the sequence of the customers in a route. Below, we only discuss how this update is performed when the rectangle and the disk compactness measures are applied. Updating the compactness cost for the hybrid measure simply combines the updates for the other two measures.

5.3.2.2.1 Rectangle compactness measure: First, recall that, for a route r visiting the subset of customers V_r,

$$\kappa_r^R = \min_{\theta \in \Theta} \kappa_r^R(\theta),$$

where

$$\kappa_r^R(\theta) = \min\{\mu, x_r^{Max}(\theta) - x_r^{Min}(\theta)\} \times \min\{\mu, y_r^{Max}(\theta) - y_r^{Min}(\theta)\}$$

and $x_r^{Max}(\theta)$, $x_r^{Min}(\theta)$, $y_r^{Max}(\theta)$, and $y_r^{Min}(\theta)$ are the maximum and minimum θ-rotated coordinate values of the customers visited in route r. Then, observe that inserting a new customer in route r cannot decrease the values $\kappa_r^R(\theta)$, $\theta \in \Theta$, whereas removing a customer from r cannot increase these values.

Let $\Theta^* = \{\theta \in \Theta \mid \kappa_r^R(\theta) = \kappa_r^R\}$ be the subset of rotation angles in Θ yielding a rectangle of minimal area.

— When a customer $v \in V^C \setminus V_r$ with θ-rotated coordinate values (x_v^θ, y_v^θ), $\theta \in \Theta$, is inserted in route r to yield a new route r', it is easy to show that $\kappa_{r'}^R(\theta) = \kappa_r^R(\theta)$ for all angles $\theta \in \hat{\Theta}$, where $\hat{\Theta} = \{\theta \in \Theta \mid x_v^\theta \in [x_r^{Min}(\theta), x_r^{Max}(\theta)] \text{ and } y_v^\theta \in [y_r^{Min}(\theta), y_r^{Max}(\theta)]\}$. Therefore, two cases can occur:

  - If $\hat{\Theta} \cap \Theta^* \neq \emptyset$, then $\kappa^R_{r'} = \kappa^R_r$.
  - Otherwise, in the worst case, the values $\kappa^R_{r'}(\theta)$ must be computed for all $\theta \in \Theta \setminus \hat{\Theta}$ to determine $\kappa^R_{r'}$, requiring $\mathcal{O}(|\Theta||V_r|)$ operations.
- When a customer $v \in V_r$ with θ-rotated coordinate values (x^θ_v, y^θ_v), $\theta \in \Theta$, is removed from route r to yield a new route r', it is easy to show that $\kappa^R_{r'}(\theta) = \kappa^R_r(\theta)$ for all angles $\theta \in \tilde{\Theta}$, where $\tilde{\Theta} = \{\theta \in \Theta \,|\, x^\theta_v \in]x^{Min}_r(\theta), x^{Max}_r(\theta)[\text{ and } y^\theta_v \in]y^{Min}_r(\theta), y^{Max}_r(\theta)[\,\}$. Again, two cases can occur:
  - If $\tilde{\Theta} = \Theta$, then $\kappa^R_{r'} = \kappa^R_r$.
  - Otherwise, in the worst case, the values $\kappa^R_{r'}(\theta)$ must be computed for all $\theta \in \Theta \setminus \tilde{\Theta}$ to determine $\kappa^R_{r'}$, taking $\mathcal{O}(|\Theta||V_r|)$ operations.

Consequently, for the rectangle measure, the worst-case complexity of updating the compactness cost of a single route r in a tabu search iteration is given by $\mathcal{O}(|\Theta||V_r||V^C|)$, *i.e.*, $\mathcal{O}(|\Theta||V_r|)$ operations for each potentially inserted or removed customer in V^C.

5.3.2.2.2 Disk compactness measure: Recall that, for a route r,

$$\kappa^D_r = \left(\frac{D_r}{2}\right)^2 \pi,$$

where $D_r = \max_{(v_i,v_j)\in V_r^2} e_{v_i,v_j}$. Let $V^* = \{v^*_1, v^*_2\}$ be the set containing the two customers in V_r defining the diameter D_r, i.e., $D_r = e_{v^*_1,v^*_2}$.

- When inserting a customer $v \in V^C \setminus V_r$ in route r to yield a new route r', we compute $D_{r'} = \max\{D_r, \max_{v'\in V_r} e_{v,v'}\}$ in $\mathcal{O}(|V_r|)$ and set $\kappa^D_{r'} = \left(\frac{D_{r'}}{2}\right)^2 \pi$.
- When removing a customer $v \in V_r$ from route r to yield a new route r', two cases can occur:
  - If $v \notin V^*$, then $\kappa^D_{r'} = \kappa^D_r$.
  - Otherwise, the diameter $D_{r'}$ needs to be computed from scratch as $D_{r'} = \max_{(v_i,v_j)\in V^2_{r'}} e_{v_i,v_j}$ in $\mathcal{O}(|V_r|^2)$.

Thus, in a tabu search iteration, updating the compactness cost of a single route r under the disk measure can be performed in a complexity of $\mathcal{O}(|V^C \setminus V_r||V_r|) + \mathcal{O}(|V_r|^2)$ because only the two customers v^*_1 and v^*_2 induce a complete calculation when a customer in V_r is checked for removal.

5.3.2.2.3 Comparison with the exact compactness measure: If the exact compactness measure was used to compute the compactness costs in the LNS heuristic, the

compactness cost $\kappa^E_{r'}$ of a route r' (*i.e.*, the area of its convex hull) obtained from a current route r by inserting a customer $v \in V^C \setminus V_r$ or deleting a customer $v \in V_r$ could be computed as follows in the tabu search algorithm. First, we check if v is located inside the convex hull of route r using the ray casting algorithm [82], which works in $\mathcal{O}(h)$, where h is the number of edges of the hull ($h = |V_r|$ in the worst case). Then, two cases can happen:

— If customer v is inside the convex hull of route r, then $\kappa^E_{r'} = \kappa^E_r$.

— Otherwise, the convex hull of route r' must be computed and its area $\kappa^E_{r'}$ computed as explained in Section 5.2.1.1. In the worst case, these computations require $\mathcal{O}(|V_r| \log |V_r|)$ operations.

Consequently, updating the compactness costs resulting from the potential insertion or removal of all customers from a single route r in a tabu search iteration has a worst-case complexity of $\mathcal{O}(|V^C||V_r| \log |V_r|)$. This complexity is clearly larger than that of the disk measure. It is also larger than that of the rectangle measure when $|\Theta| < \log |V_r|$. Note, however, that with the exact measure, $\mathcal{O}(|V^C||V_r|)$ operations are mandatory to check whether the customers are inside the convex hull of the current route and determine if additional computations are necessary for each customer, whereas with the rectangle measure, $\mathcal{O}(|V^C||\Theta|)$ operations are required to assess this. Given that, for our tests, $|\Theta|$ is much less than $|V_r|$, we believe that the practical complexity of updating the compactness costs is much smaller with the rectangle measure than with the exact measure. In fact, our test results show that the rectangle measure performs either better or very similar to the disk measure.

5.4 Computational results on academic instances

As mentioned in the introduction, our goal is not to evaluate the performance of the proposed LNS heuristic, but rather to compare the compactness measures and analyze their impact on the solution characteristics. To a lesser extent, we also evaluate the impact of using these compactness measures on the computational time of our heuristic.

In this section, we present the computational results obtained on academic benchmark instances derived from the VRPTW instances of [83] which will be referred to as the GH instances. The size of these instances ranges from 200 to 1000 customers but we limited our tests to those involving 200 and 400 customers. For each size, there are three groups of 20 instances: Clustered (C), Random (R) and Random-Clustered (RC), which is a mix between the first two categories. Because the C group is not very representative of postal services' instances and not relevant from a compactness point of view (most optimal routes are compact without incentives), our tests have focused on the two last categories (R & RC). Moreover, each group is divided according to two types of time horizons: short and long. In

the short-horizon instances, a single vehicle can visit around 10 customers, which is far below the average for postal services. Therefore, we retained the instances of the groups R2 and RC2 (with about 50 customers per vehicle). The 40 selected VRPTW instances (20 with 200 customers and 20 with 400 customers) were simply converted into Comp-VRPTW by considering the compactness costs. Because it is not straightforward to assign a practical value to the maximum route duration U^{max} parameter for these instances, the maximum duration constraint has been omitted. This allows us to compare the computed Comp-VRPTW solutions and the best-known VRPTW solutions reported on the [84] website. The latter solutions are referred to as the *TT-BKS* to highlight that they seek to minimize the total travel time. Note that, in the GH instances, the travel times are equal to the Euclidian travel distances and that, for our tests, the travel times are truncated to the second decimal digit before being multiplied by 100.

Below, we only present results obtained by varying certain parameter values impacting only the TDR phase. We exclude from these results and the discussions the VNR phase which is assumed pre-executed, *i.e.*, for each instance, the starting point of each TDR phase is the same independently of the TDR parameter setting. Unless otherwise specified, the LNS heuristic executes 50 LNS iterations in the TDR phase. It removes 70 customers (for the 200-customer instances) or 100 (for the 400-customer ones) at each iteration. When the rectangle or hybrid compactness measure is used, the set of rotation angles Θ is defined according to a discretization of the interval $[0, 90[$. By default, we chose to discretize it with an increment of $\gamma = 15$, *i.e.*, $\Theta(\gamma) = \{0, 15, 30, 45, 60, 75\}$.

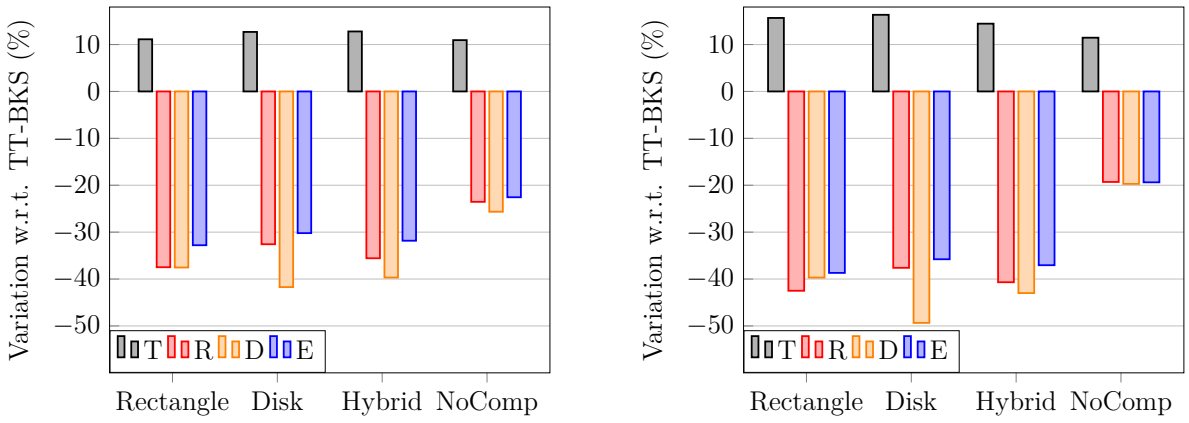
5.4.1 Comparison of compactness measures

We start by comparing the compactness measures. All 40 instances were solved using four variants of the LNS heuristic which only differ by the compactness measure used: Rectangle ($\lambda^R = k, \lambda^D = 0$), Disk ($\lambda^R = 0, \lambda^D = k$), Hybrid ($\lambda^R = \lambda^D = k/2$), and NoComp ($\lambda^R = \lambda^D = 0$), where k is set to 3. With such parameters, the compactness cost of the solution obtained is approximately 20% of the total travel cost. A sensitivity analysis on the value of λ^R when applying the Rectangle variant will be presented in Section 5.4.3. Note that the NoComp variant does not consider the compactness features. Each computed solution is evaluated *a posteriori* with respect to all compactness measures, including the exact measure (based on the route convex hull). For example, the solution of an instance that was computed using the rectangle compactness measure in the LNS heuristic is evaluated at the end of the optimization process using the disk, hybrid, and exact measures. Furthermore, the four compactness measures are computed for all *TT-BKS*.

The results of these tests are summarized in Figures 5.7a and 5.7b for the 200- and 400-customer instances, respectively. For each algorithm variant (identified on the horizontal axis by the compactness measure used), the four vertical bars indicate the average variation in percentage of the travel (routing) cost (T, black), the rectangle (R, red), disk (D, orange), and exact (E, blue) compactness costs of the computed solution with respect to the *TT-BKS*. For example, the set of four bars above Disk in Figure 5.7a show that, compared to the *TT-BKS*, the travel cost of the solution computed by the Disk variant of the LNS algorithm is on average 12.7% larger but that the rectangle, disk, and exact compactness costs reduce by 32.6%, 41.7%, and 30.2%, respectively. Note that the travel cost excludes any vehicle fixed cost and that for some 400-customer instances, our LNS algorithm was unable to reach in the VNR phase the minimum number of vehicles used in the *TT-BKS*.

For both instance sets, the improvement in the compactness costs is significant, varying from 32.6% to 49.4% depending on the instances and the measure used in the algorithm. The final improvement of the exact measure value is also quite steady, ranging from 30.2% to 38.7%. The tradeoff is achieved by increasing the travel cost by a substantial amount (between 11.1% and 16.3%). As the NoComp results show, our method (especially when starting from an initial solution favoring compactness and using such a low number of LNS iterations) cannot compete with state-of-the-art algorithms to minimize the travel time only. However, when dealing with compactness, the tradeoff is unavoidable as it will be further discussed in Section 5.4.3. Comparing the compactness results of the first three algorithm variants to the NoComp variant, we observe that much larger improvements are obtained when compactness is considered in the algorithm.

Concerning the compactness measures used in the LNS heuristic, we do not observe a large difference on the average variation of the exact measure value (between 30.2% and 32.8% for the 200-customer instances and between 35.8% and 38.7% for the 400-customer instances) they induce. Nevertheless, the rectangle measure yields the largest average improvement for both instance sets, whereas the disk measure is the less effective although it succeeds to reduce the disk measure value the most. To further compare the compactness measures, Table 5.2 reports the average computational time (CPU) required in the TDR phase by each algorithm on each set of instances as well as the average time increase (Δ CPU) with respect to the time required by the NoComp algorithm variant. These results show that dealing with compactness can moderately increase the average computational times of the LNS heuristic. This increase is larger in proportion for the 200-customer instances because solving the restricted master problems requires much more time for the 400-customer instances. Furthermore, these results indicate that the rectangle compactness measure is less time-consuming than the disk measure and the hybrid measure which combines the other two. Consequently,



(a) 200 customers

(b) 400 customers

Figure 5.7 Comparison of compactness measures

it appears that the rectangle measure offers the best performance in terms of both route compactness quality and computational time.

5.4.2 Sensitivity to rotation angle discretization

To evaluate the sensitivity of the rectangle measure to the angle discretization increment γ as defined in the previous section, we ran tests with the Rectangle variant ($\lambda^R = 3, \lambda^D = 0$) of the LNS heuristic using different γ values, namely, $\gamma = 2, 5, 15, 30, 45, 90$. For each instance and each γ value, we obtained a solution which was evaluated *a posteriori* according to three different compactness measures: $\sum_r K_r^R(\Theta(\gamma))$, $\sum_r K_r^R(\Theta(1))$, and $\sum_r \kappa_r^E$, where the sums are taken over the routes in the solution and $K_r^R(\Psi) = \min_{\psi \in \Psi} \kappa_r(\psi)$. The first measure is the rectangle measure used by the algorithm, i.e., with γ to determine the rotation angle set Θ. The second is the rectangle measure with a discretization increment $\gamma = 1$ which allows to estimate the error committed by a coarser discretization. Finally, the last one is the exact compactness measure which allows to determine the error induced by replacing the area of the route convex hull by the area of the best rotated rectangle.

The results of these experiments are presented in Figure 5.8 which is divided in two parts according to the size of the instances. The red, green and blue curves indicate the average compactness costs in function of γ for the above three measures, respectively. The black curve illustrates the average increase in the TDR phase computational time with respect to the time obtained for the no rotation case ($\gamma = 90$). Rationally, one can expect that the lower the value of γ, the smaller the difference between $\sum_r K_r^R(\Theta(\gamma))$ and $\sum_r K_r^R(\Theta(1))$, and

Table 5.2 Average computational times for the GH instances

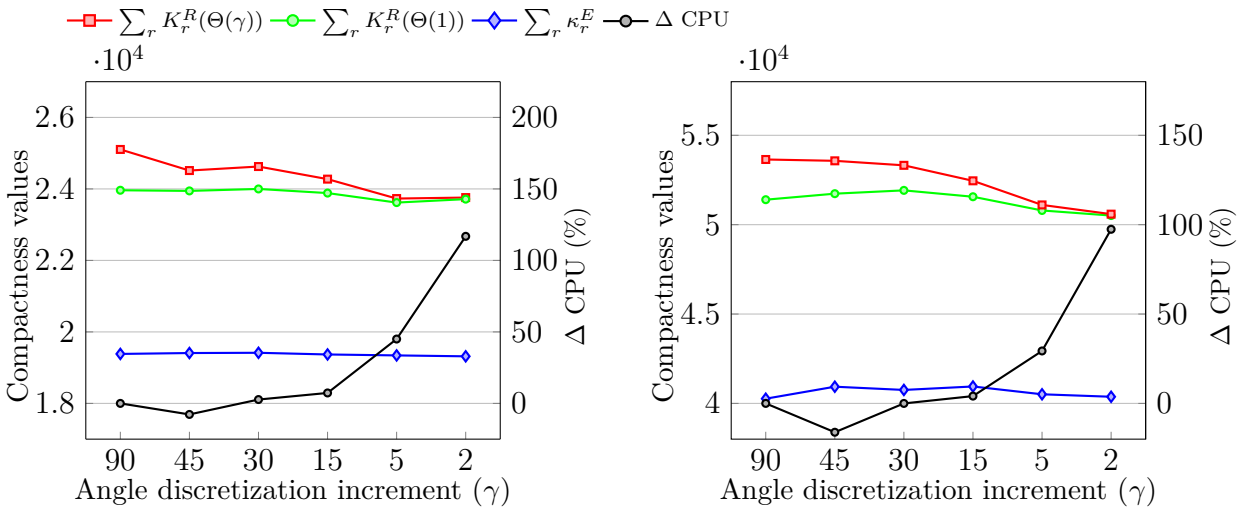
	200 customers		400 customers	
Variant	CPU (s)	Δ CPU (%)	CPU (s)	Δ CPU (%)
NoComp	127	–	455	–
Rectangle	188	48.0	517	13.6
Disk	191	50.4	588	29.2
Hybrid	231	81.9	665	46.2

the longer it takes to execute the 50 LNS iterations in the TDR phase. In practice, these tendencies are observed but not with a perfect monotonicity. On the other hand, the impact of decreasing the value of γ on the exact measure value $\sum_r \kappa_r^E$ is not significant on average.

To pursue our analysis, we extracted additional statistics from the solutions obtained with different γ values, but also the solutions produced by the Disk variant and the Hybrid variant (with $\gamma = 15$) of the LNS heuristic. More precisely, for each algorithm, instance and route in the solution obtained, we computed the relative error between the compactness costs evaluated according to the measure used in the algorithm and the exact measure. For each instance size, Table 5.3 reports the average of these relative errors (Err.) and the standard deviation (σ). With regards to the rectangle measure, we observe a substantial gain when considering rotated rectangles, even with a single 45°-rotation ($\gamma = 45$). Taking into account the Δ CPU values of Figure 5.8, a value of $\gamma = 15$ seems to be a good tradeoff between measure accuracy and computational time. For the disk and hybrid measures, the errors are much larger than with the rectangle measure. In fact, we have noticed that, in general, the disks do not fit the routes as well as the rectangles. In addition, we have observed some pathological routes that are much better approximated by a rectangle with a large length compared to its width than by a disk.

5.4.3 Sensitivity to rectangle measure weight

In this section, we evaluate the sensitivity of the value of the weight parameter λ^R that is used to put more or less emphasis on reducing the compactness cost with respect to the travel cost when the Rectangle variant ($\lambda^R > 0$, $\lambda^D = 0$) of the LNS heuristic is applied. To do so, we solved all 40 GH instances using different λ^R values, *i.e.*, $\lambda^R \in \{0.5, 1, 3, 5, 10, 20, 50\}$. The plots in Figure 5.9, for 200- and 400-customer instances, show the average variations (with respect to the TT-BKS) of the compactness costs and the travel (routing) costs in function of the weight λ^R, where its value increases from left to right. The blue curve is associated with the exact compactness measure whose value is computed *a posteriori*, whereas the red curve refers to the rectangle measure, with $\gamma = 15°$. Each curve defines an approximate



(a) 200 customers

(b) 400 customers

Figure 5.8 Sensitivity of the compactness costs and computational time to the value of γ

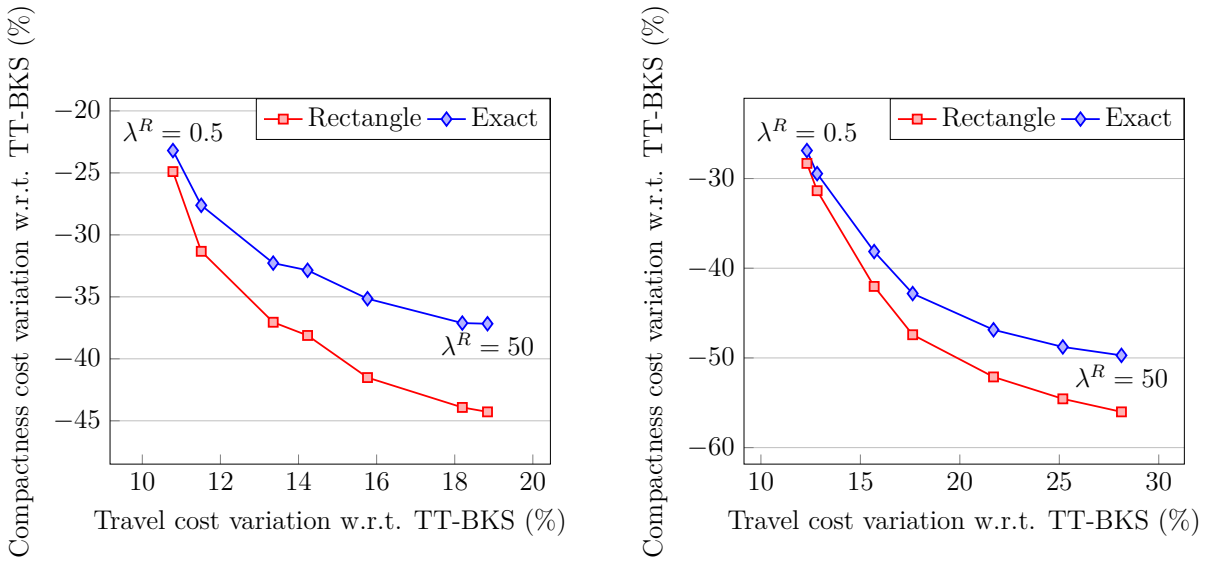
Table 5.3 Relative error with respect to the exact measure and its standard deviation

		200 customers		400 customers	
Measure	γ	Err. (%)	σ (%)	Err. (%)	σ (%)
Rectangle	90	28.9	20.3	34.9	20.5
	45	25.8	10.5	31.6	13.2
	30	26.3	10.8	31.9	12.4
	15	23.8	9.1	29.5	11.0
	5	22.8	9.6	26.5	10.4
	2	21.0	8.5	26.1	11.3
Disk		49.7	19.7	50.1	31.1
Hybrid	15	41.6	16.9	52.0	25.8

Pareto-front. The larger the λ^R value, the better the improvement in the compactness cost, but the larger the travel cost. As a reference to evaluate the incurred tradeoff, when $\lambda^R = 1$, the compactness cost is worth about 5% to 10% of the travel cost.

5.4.4 Impact of initial solution

In this section, we discuss the impact of using the initial solution computed by the approach presented in Section 5.3.1 and referred to as IS below. In fact, we compare its usage with using the TT-BKS as the initial solution. Starting from the TT-BKS, all 40 GH instances were, thus, solved again using the Rectangle variant of the LNS heuristic and the different λ^R values specified in Section 5.4.3. Figure 5.10 provides two tradeoff curves between the average



(a) 200 customers

(b) 400 customers

Figure 5.9 Sensitivity of the compactness and travel costs to the value of λ^R

variations (with respect to the TT-BKS) of the compactness cost (computed using the exact measure) and the travel cost in function of λ^R. This time, the averages are computed over the 200- and 400-customer instances together. The blue curve is obtained when starting the LNS heuristic from IS (therefore, it corresponds to the average of the two blue curves in Figure 5.9), whereas the gray one arises when it begins from TT-BKS.

Recall that the procedure used to compute IS favors compactness. This is confirmed by Figure 5.10, where one can see that starting from IS enables to find very compact final solutions even with a small weight λ^R. With larger weights, these starting points generate the best solutions relatively to the compactness criterion in the limited number of LNS iterations performed. However, if a large travel cost increase cannot be afforded to improve compactness, it seems better to start from an initial solution that has a travel cost close to that of the TT-BKS.

5.4.5 Examples of compact and non-compact solutions

In this section, using solutions of the RC2_2_3 instance with 200 customers, we show how compact and non-compact solutions can look like, and how the rectangle and disk compactness measures can be interpreted visually. In Figures 5.11a and 5.11b, we present the TT-BKS and the solution obtained by the Rectangle variant of the LNS heuristic, respectively (the black square represents the depot). Both solutions comprise four routes that are

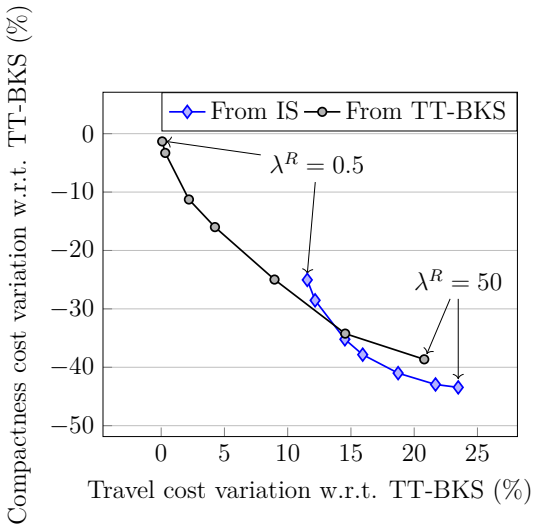


Figure 5.10 Initial solution impact with respect to varying λ^R values

highlighted with different colors. For each route, the corresponding rectangle is the one used to compute its compactness and the shaded polygon corresponds to its convex hull. For the TT-BKS, we obtain the following statistics:

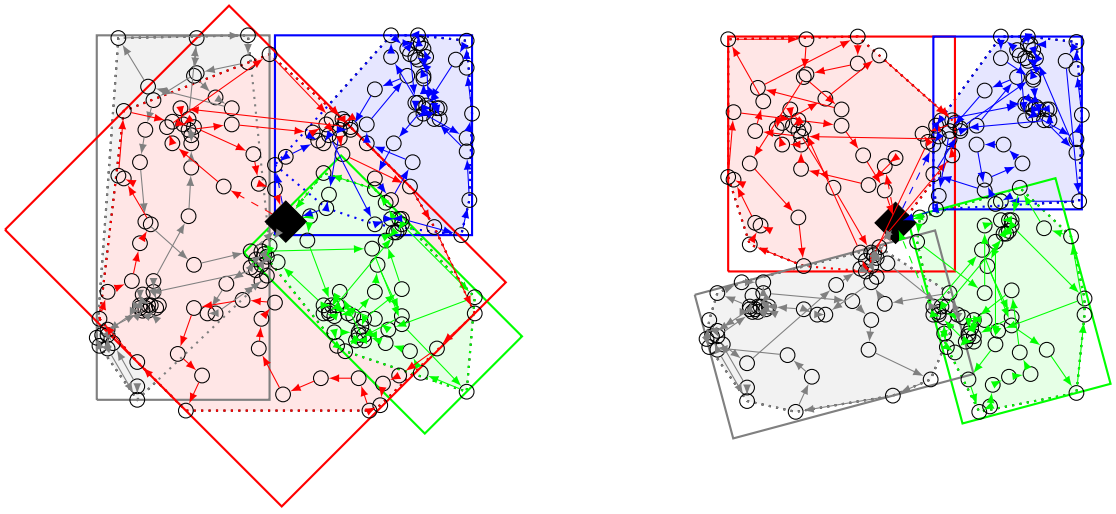
$$\sum_{r\in\Omega^B} \tau_r = 260186, \quad \sum_{r\in\Omega^B} \kappa_r^E = 27025, \quad \sum_{r\in\Omega^B} \kappa_r^R = 35881, \quad \min_{r\in\Omega^B} \kappa_r^R = 4824, \quad \max_{r\in\Omega^B} \kappa_r^R = 17015,$$

where Ω^B is the set of routes in the TT-BKS. For the Rectangle variant heuristic solution composed of the set of routes Ω^R, we get:

$$\sum_{r\in\Omega^R} \tau_r = 290990, \quad \sum_{r\in\Omega^B} \kappa_r^E = 16506, \quad \sum_{r\in\Omega^R} \kappa_r^R = 20379, \quad \min_{r\in\Omega^R} \kappa_r^R = 3520, \quad \max_{r\in\Omega^R} \kappa_r^R = 7308.$$

Observe first that, compared to the TT-BKS, the travel cost of the Rectangle variant solution increases by close to 12% but the total exact (resp., rectangle) compactness cost decreases by 43% (resp., 39%). In both solutions, the two routes on the right (blue and green) cover each approximately the same area in both solutions, but the other two routes differ significantly from one solution to the other. In fact, one can see that the red route in the TT-BKS travels almost all around the whole region (except the top-right corner) and induces a large compactness cost. Observe also that route overlapping is very limited in the compact Rectangle variant solution even if this criterion is not taken into account in the objective function.

Similar to Figures 5.11a and 5.11b, Figures 5.12a and 5.12b allow to compare the TT-BKS



(a) TT-BKS (b) Rectangle variant solution

Figure 5.11 Structure of the TT-BKS and the Rectangle variant solution

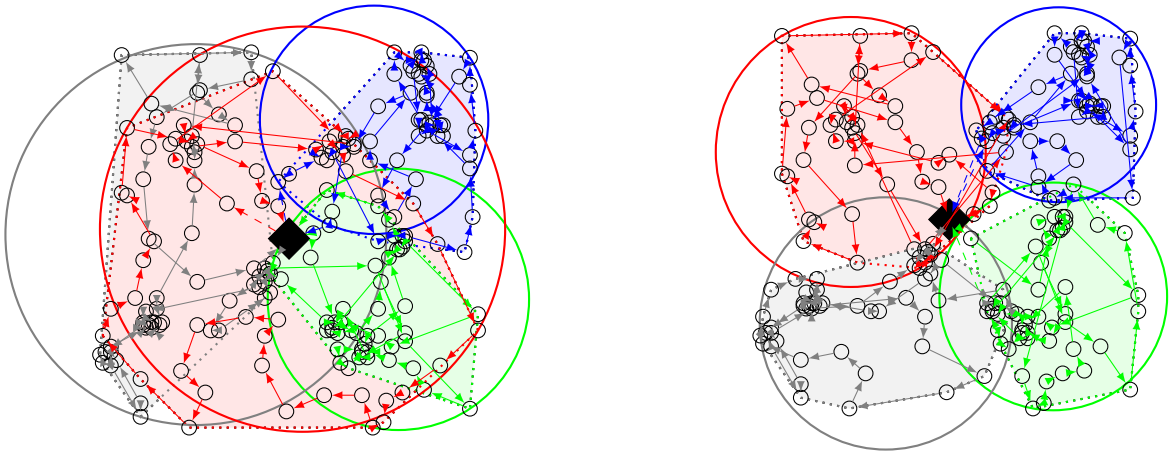
with the solution computed by the Disk variant LNS heuristic, using the disk measure to evaluate compactness. The TT-BKS exhibits the following statistics:

$$\sum_{r\in\Omega^B} \tau_r = 260186, \quad \sum_{r\in\Omega^B} \kappa_r^E = 27025, \quad \sum_{r\in\Omega^B} \kappa_r^D = 46306, \quad \min_{r\in\Omega^B} \kappa_r^D = 5621, \quad \max_{r\in\Omega^B} \kappa_r^D = 17691,$$

whereas the Disk variant heuristic solution, with a set of routes Ω^D, has the following ones:

$$\sum_{r\in\Omega^D} \tau_r = 286372, \quad \sum_{r\in\Omega^B} \kappa_r^E = 16457, \quad \sum_{r\in\Omega^D} \kappa_r^D = 24387, \quad \min_{r\in\Omega^D} \kappa_r^D = 4092, \quad \max_{r\in\Omega^D} \kappa_r^D = 7852.$$

Here, we observe a travel cost increase of approximately 10% to obtain a large 39% decrease of the total exact compactness measure (47% of the disk measure). This decrease is more substantial than the one reported above for the rectangle measure. This is due to the fact that, for this example, the disk measure is less accurate than the rectangle measure and greatly overestimates the area of the convex hull for certain routes of the TT-BKS, especially for the leftmost route (in gray). This higher inaccuracy induces a large difference between the compactness costs (areas of rectangles or disks) of the Rectangle and Disk variant solutions (20379 and 24387) although these solutions are quite similar.



(a) TT-BKS (b) Disk variant solution

Figure 5.12 Structure of the TT-BKS and the Disk variant solution

5.5 Computational results on instances derived from a real-world dataset

Our industrial partner Giro Inc., which commercializes optimization software to postal agencies, gave us one real-world Comp-VRPTW instance with 1553 customers, including only 16% with a time window. Beside the customer time windows, the data specify the estimated travel time between each pair of locations, the service time at each customer, and the geographical coordinates of each customer location, allowing us to compute the Euclidian distance between each pair of customers. From this instance, we extracted subsets of customers randomly to create a total of 12 instances, namely, three instances of each of the following sizes: 200, 300, 400 and 500 customers. Our computational experiments have focussed on these smaller-sized instances as our LNS heuristic is not well suited to solve the large original instance. Tackling it would require the development of acceleration strategies and is, thus, left for future work.

These 12 instances are hereafter called the industry-derived instances and some of their characteristics are provided in Table 5.4. For each instance size (# Customers) and each of the three instances of this size (denoted $I1$, $I2$ and $I3$), this table indicates the number of customers with a time window (# TWs) and the number of vehicles (# Vehicles) used in the solution obtained at the end of the VNR phase of the LNS heuristic. One can observe that the percentage of customers with a time window varies between 12% and 19.5% and that the average number of customers per vehicle ranges between 60 and 80. For all these instances, the maximum route duration U^{max} is set to 8 hours.

Each industry-derived Comp-VRPTW instance was solved four times using the four variants of the proposed branch-and-price-based LNS heuristic: Rectangle ($\lambda^R = 2k$, $\lambda^D = 0$), Disk

Table 5.4 Characteristics of the industry-derived instances

	# TWs			# Vehicles		
# Customers	$I1$	$I2$	$I3$	$I1$	$I2$	$I3$
200	29	26	39	3	3	3
300	36	46	58	4	4	5
400	63	54	55	6	5	5
500	82	78	81	7	7	7

($\lambda^R = 0$, $\lambda^D = k$), Hybrid ($\lambda^R = k$, $\lambda^D = 0.5k$), and NoComp (without considering compactness, i.e., $\lambda^R = \lambda^D = 0$), where k is a predefined constant such that the total compactness cost of a compact solution represents about 20% of its total travel cost. Note that less weight is put on the disk measure compared to the rectangle measure because the former tends to overestimate more the exact measure as shown by the results in Table 5.3. When the rectangle or hybrid compactness measure is used, the rotated angle discretization increment is set to $\gamma = 15$. In all cases, the initial solutions are computed as explained in Section 5.3.1. For instances with 200 customers, the set of customers is divided into 4 subsets, whereas, for the other instances, 9 subsets are used. As in the previous section, the VNR phase is the same for all heuristic variants, while the TDR phase differs according to the compactness measure applied. For all variants, 100 LNS iterations are performed in the TDR phase.

In the following, we report computational results where the compactness is defined using geographical areas (Section 5.5.1) and using maximum travel times (Section 5.5.2).

5.5.1 Compactness based on geographical areas

As in Section 5.4.1, we compare the routing/compactness cost tradeoff induced by the solutions obtained with each algorithm variant taking into account compactness when measured using geographical areas, as before. The results for the industry-derived instances are summarized in Figure 5.13 which is crafted as a subfigure of Figure 5.7, except that the variations are computed with respect to the NoComp variant solutions (instead of the TT-BKS).

From these results, we observe again that taking into account compactness during optimization (Rectangle, Disk and Hybrid variants) can yield much more compact routes, with an average gain of the exact measure varying between 9.7% and 11.3%. The travel cost results show that minimizing compactness helps to further diminish the travel costs (average reductions varying between 0.2% and 1.1%). This can be explained by the fact that, when the percentage of customers with a time window is small as in the industry-derived instances, routes minimizing the total travel cost naturally tend to be compact and, consequently, favoring compactness also favors shorter routes. Comparing the performance of the three heuristic

variants considering compactness, one can see that the Disk variant yields slightly better solutions with respect to the exact compactness measure and the travel costs, even though the disk measure does not seem as reliable as the rectangle and hybrid measures.

In Table 5.5, we report the average computational time (CPU) required in the TDR phase by each algorithm variant as well as the average time increase (Δ CPU) compared to the reported NoComp time. These results indicate again that handling the compactness costs increases the computational times of our LNS heuristic moderately (between 25.5% and 34.2% on average). This increase is, however, less important (especially for the Hybrid variant) than for the academic instances (see Table 5.2), probably because the objectives of minimizing compactness costs and of minimizing travel costs are less conflicting when there are less time windows as mentioned above.

Table 5.5 Average computational times for the industry-derived instances

Variant	CPU (s)	Δ CPU (%)
NoComp	509.2	–
Rectangle	664.7	30.6
Disk	643.9	26.5
Hybrid	683.2	34.2

5.5.2 Compactness based on maximum travel times

We mentioned in Section 5.2.1.3 that other measures than areas based on Euclidean distances could be used to evaluate the compactness of a route. Using travel time is particularly useful when Euclidean distances do not accurately reflect the road network. In this section, we propose to determine if the disk measure where the diameter D_r of a route is defined by the maximum travel time between two customers in this route can help reducing the geographical compactness cost of the computed routes. Hereafter, the heuristic variant using this travel time disk measure is called the Disk-TT variant.

All industry-derived instances where solved using the Disk-TT variant. A posteriori, we computed the geographical exact compactness measure of each route in each computed solution. Furthermore, the solutions produced by the Rectangle, Disk and NoComp variants were also evaluated a posteriori with respect to the travel time disk compactness measure. Figure 5.14 allows to compare the Rectangle, Disk and Disk-TT variant solutions with respect to the travel costs (T), the travel time disk compactness costs (DTT), and the geographical exact compactness costs (E). Again, the bars indicate the average variations of these costs with respect to the costs incurred by the NoComp variant solutions. The results show that the Disk and Disk-TT variants perform similarly with respect to the travel time disk measure.

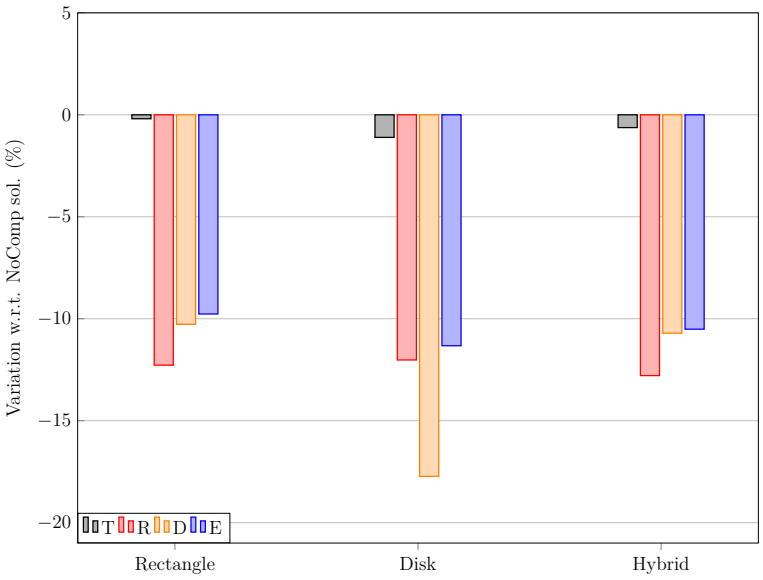


Figure 5.13 Comparative results for industry-derived instances

However, on average, the Disk-TT variant solutions have smaller total travel costs but larger geographical exact compactness costs. This is not surprising because minimizing the travel time disk measure can avoid assigning to the same route pairs of customers that are geographically close but not easily linkable by the road network. As for the Rectangle variant, it generates solutions that are not competitive with the other variants when considering the travel time disk measure because this variant does not necessarily disadvantage routes that cover areas which are longer than wider, inducing a larger maximum travel time disk measure. Finally, for the Disk-TT variant, we report an average computational time of 787.9 seconds, which is an increase of 54.7% compared to the NoComp variant. This increase is somewhat larger than those observed for the other variants (see Table 5.5).

5.6 Conclusion

In this paper, we proposed two compactness measures to favor generating compact routes when solving the Comp-VRPTW. Considering the exact compactness measure of a route as the area of its convex hull, we approximated this measure by replacing the convex hull by either a rectangle or a disk. When geographical measures based on Euclidean distances are not suitable due to the actual road network, the disk approximation can be adapted by redefining the disk diameter as, e.g., the maximum travel time between two customers in the route. Next, to take into account the minimization of these compactness measures, including

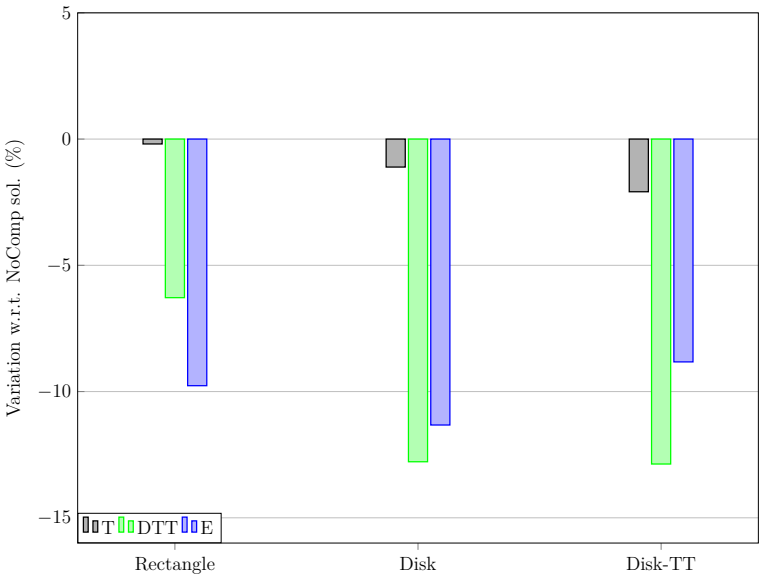


Figure 5.14 Comparative results involving the Disk-TT variant

a hybridization of the rectangle and disk measures, we modified the branch-and-price-based LNS heuristic of [50]. Our computational experiments on academic Comp-VRPTW instances with time windows for all customers showed that the proposed heuristic variants can greatly reduce the compactness costs by around 35% in exchange of increasing the travel costs by more than 10%. On the other hand, for industry-derived instances involving up to 500 customers with less than 20% subject to a delivery time window, favoring compactness during the optimization achieved a gain of about 10% on the compactness costs while also slightly improving the travel costs. In all cases, handling compactness increases the average computational times by 13% to 81%. Finally, we have seen that when using travel time instead of Euclidean distance to define the diameter in the disk measure, it provides some improvement in maximum-time-travel compactness but also in geographical compactness.

As future work, we want to develop speedup strategies for the proposed LNS heuristic for efficiently tackling real-life instances containing several thousands customers.

Acknowledgement: We would like to thank Charles Fleurent and his team at GIRO Inc. for suggesting this problem to us and providing a real-life dataset. We gratefully acknowledge the financial support of GIRO Inc. and the Natural Sciences and Engineering Research Council of Canada under the grant RDCPJ 520349-17.

5.A Appendix: Constraint programming model used to compute an initial solution

To compute an initial solution for the proposed branch-and-price-based LNS heuristic, we use the CPLEX constraint programming optimizer [CP-Optimizer, for additional details see 85] and apply it on the following model which we describe using the CP-Optimizer variables and constraints. First, for each customer $i \in V^C$, we define a master task as a mandatory interval variable A_i and its optional copies B_i^l for each allowed vehicle $l \in L$. For each vehicle l, we also add extra interval variables $\underline{D}^l$ and $\overline{D}^l$ representing the depot at the start and the end of its route. Earliest start time, latest start time and length are predefined attributes of interval variables, which are used to model the time window and service time (if any) at each customer. A route assigned to vehicle l is then defined by a SequenceVariable Seq^l involving $\left(\overline{D}^l \cup (\cup_{i \in V^C} B_i^l) \cup \underline{D}^l\right)$. Denoting by T_X the start time of a task X and by $[\![\chi]\!]$ the boolean value of expression χ, the constraint programming model is as follows:

$$\min \quad (\sum_{l \in L} T_{\overline{D}^l} - T_{\underline{D}^l}) + F \times \sum_{l \in L} [\![T_{\overline{D}^l} - T_{\underline{D}^l} > 0]\!] \tag{5.14}$$

$$\text{subject to:} \qquad Seq^l.NoOverlap(\Upsilon), \qquad \forall l \in L \tag{5.15}$$

$$Seq^l.First(\underline{D}^l), \qquad \forall l \in L \tag{5.16}$$

$$Seq^l.Last(\overline{D}^l), \qquad \forall l \in L \tag{5.17}$$

$$T_{\overline{D}^l} - T_{\underline{D}^l} \leq U^{max}, \qquad \forall l \in L \tag{5.18}$$

$$Alternative(A_i, \cup_{l \in L} B_i^l), \qquad \forall i \in V^C. \tag{5.19}$$

Objective function (5.14) seeks to minimize the sum of the route durations and the fixed cost for each vehicle used. Indeed, $T_{\overline{D}^l} - T_{\underline{D}^l} > 0$ when at least one customer is visited in the route selected for vehicle l. Given the transition distance matrix Υ, the *NoOverlap* constraints (5.15) ensure that there is sufficient time to travel between two customers visited consecutively in a route. To properly define the sequences, constraints (5.16) and (5.17) impose that each route starts and ends at the depot, respectively. The partitioning of the customers $i \in V^c$ in the different vehicles l is handled by the *Alternative* constraints (5.19) on A_i and B_i^l. These constraints work as follows: if A_i is present (which is mandatory in our case), then exactly one variable in $\cup_{l \in L} B_i^l$ must also be present. This forces each customer to be visited once and only once by a single vehicle.

CHAPITRE 6 ROUTES COMPACTES POUR LES GRANDES INSTANCES DE SERVICES POSTAUX

6.1 Introduction

Après avoir attaqué le problème de routes compactes pour la livraison de colis dans les services postaux avec des instances de petite et moyenne taille (jusqu'à 500 clients), nous proposons dans ce chapitre de voir les ajustements nécessaires à la résolution d'instances de taille industrielle, comptant parfois plus de 3000 clients. Il s'agit d'une adaptation de la LNS présentée dans le chapitre précédent (voir notamment figure 5.5) portant sur deux axes principaux. Le premier consiste à accélérer la résolution des sous-problèmes en limitant le voisinage de chaque route de façon dynamique. Le second a pour objectif principal d'intensifier la destruction des solutions dans diverses régions de l'instance en partitionnant itérativement l'instance afin de rendre la méthode globale plus efficace grâce à des résolutions parallèles. Dans ce chapitre, comme dans le précédent, la qualité des routes sera évaluée à travers le coût de transport et la mesure de compacité de l'ensemble des routes, tout en gardant à l'esprit de garder le temps d'exécution le plus faible possible. Compte tenu de la taille des instances, ce chapitre ne traitera que de la phase TDR (*Traveled Distance Reduction*) du LNS originel [50], les solutions initiales étant fournies par Giro, notre partenaire industriel. Le nombre de véhicules à utiliser correspond alors à celui défini par ces solutions. Le problème en tant que tel et les caractéristiques du milieu postal demeurent cependant les mêmes que dans le chapitre précédent.

Dans ce chapitre, nous présenterons tout d'abord nos travaux effectués sur la recherche de bons voisinages autour de nos routes (section 6.2). Dans la section 6.3, nous appréhenderons un mécanisme de séquencement de partitions mis en place pour intensifier la force du LNS dans diverses zones géographiques de l'instance. Enfin, nous présenterons les résultats obtenus sur les instances industrielles à notre disposition dans la section 6.4 avant de conclure.

6.2 Réduction des voisinages

Pour résoudre des instances de taille conséquente (1000-4000 clients) dans un temps convenable, nous avons dû adapter la méthode utilisée dans l'article présenté précédemment (Chapitre 5). Rappelons que dans chaque itération LNS, l'algorithme tente d'améliorer la solution courante en détruisant d'abord une partie de celle-ci avant de la réparer en appliquant un algorithme de génération de colonnes heuristique. Cet algorithme de génération de colonnes

cherche à chaque itération des routes de coût réduit négatif en appliquant un algorithme de recherche taboue. Celui-ci est lancé plusieurs fois, en partant de routes déjà générées. Pour comprendre la motivation qui nous a poussés à définir de nouveaux voisinages, il nous faut rentrer un peu plus dans les détails de la méthode taboue implémentée. Pour une route donnée, nous essayons itérativement d'ajouter tous les clients existants à toutes les positions de la route, tout en respectant bien entendu les sous-chemins fixés suite à la phase de destruction du LNS. Plus l'instance grossit, plus cet algorithme glouton consomme du temps de calcul sans forcément être très performant. Les essais d'insertion se font de manière systématique. Par conséquent, nous essayons régulièrement d'insérer (sans succès) des clients générant de grands détours. Il nous est alors apparu naturel de nous pencher vers l'agrégation de clients. De plus, dans un contexte de recherche de routes compactes, regrouper des clients selon leur proximité géographique a encore plus de sens. Parmi les méthodes d'agrégation (en anglais *clustering*), *k-means* est probablement l'approche la plus répandue. Comme dans beaucoup d'algorithmes d'agrégations, les groupes obtenus sont des sous-ensembles disjoints de l'ensemble des points de l'instance initiale. Dans notre problème, cette catégorisation très stricte des différents clients par rapport à leur situation géographique semble problématique au cours du processus. En effet, dans des VRPTW, il ne parait pas naturel de combiner routes et agrégations, ne serait-ce que par leur nombre. En définissant autant de groupes de clients que de routes, rien ne garantit que nous puissions construire des routes réalisables pour chacun d'entre eux. En plus d'éventuels problèmes de fenêtres de temps et de durée de route, il est très complexe de contrôler la taille de ces agrégats. C'est pourquoi nous avons réfléchi à des agrégations plus souples, mais aussi plus dynamiques, pour répondre à notre problématique.

6.2.1 Degrés d'appartenance

Dans ce chapitre, beaucoup de mécaniques tourneront autour de degrés d'appartenance. Nous parlerons d'appartenance d'un élément n (d'un ensemble N) à un élément a (d'un autre ensemble A), typiquement d'un client à un agrégat. Le principe s'appuie sur les travaux initiés par Bezdek *et al.* [62] relatifs aux agrégations dites "floues". Au lieu d'avoir des agrégats disjoints, on donne un degré d'appartenance $M(a, n)$ de chaque client n à chaque agrégat a, de façon à ce que $\sum_a M(a, n) = 1$ et $\forall n \in N, M(a, n) \geq 0$. Cette matrice M nous permet d'identifier des centres pour chaque agrégat. Dans un processus itératif, on pourra par la suite calculer des distances à ces différents centres et mettre à jour la matrice d'appartenance jusqu'à un certain critère d'arrêt (pseudoalgorithme 5). De manière imagée, plus un point est proche d'un centre, plus son degré d'appartenance est proche de 1, ce qui signifie par extension que son degré d'appartenance envers les autres agrégats est très faible.

Pour calculer les matrices d'appartenance, on suit un pseudoalgorithme semblable à celui décrit ci-dessous.

Algorithme 5 Degré d'appartenance

1: Définir le nombre d'agrégats souhaité $|A|$
2: Trouver une matrice d'appartenance initiale M^0
3: $k = 0$
4: **do**
5: $\quad k = k + 1$
6: $\quad$ mettre à jour les centres
7: $\quad$ mettre à jour les distances au centre
8: $\quad$ mettre à jour la matrice d'appartenance
9: **while** Condition Suffisante

M^k ($k \geq 0$) désigne la matrice M après la k^{ieme} itération. M^0 est la matrice initiale d'appartenance. Elle peut être définie à partir d'un partitionnement aléatoire ou non. Il est notamment possible de l'initialiser à partir d'une partition de k-means avec $k = |A|$. La condition de la ligne (9) peut être un nombre d'itérations, mais est plus généralement un critère de convergence dépendant d'une mesure entre M^{k+1} et M^k.

Cette notion de degré d'appartenance est très intéressante lorsqu'arrive le moment de créer des voisinages, surtout si nous souhaitons avoir de la flexibilité sur ceux-ci. En effet, il est possible de créer des voisinages pertinents selon une taille souhaitée ou un degré d'appartenance minimum. Pour un agrégat a donné, nous pouvons donc définir des voisinages considérant les X clients au plus grand degré d'appartenance, ou tous les clients au degré d'appartenance supérieur à un seuil D_{min} donné.

6.2.2 Agréger relativement aux routes

Si les degrés d'appartenance peuvent nous permettre de créer des voisinages modulables et contrôlables en taille, le fait que leurs définitions soient propres à l'instance et non pas à la solution du problème en elle-même peut être contraignant sur l'espace des solutions. Il faut aussi garder à l'esprit que nous avons des fenêtres de temps dans notre problème, qui amènent parfois des incohérences par rapport à la notion de proximité géographique.

Pour cette raison, nous avons proposé de calculer des degrés d'appartenance non pas entre les clients et des agrégats *a priori* inconnus, mais entre les clients et les routes d'une solution entière. La logique demeure la même, mais dorénavant, les centres des agrégats sont connus *a priori* car basés sur les routes de cette solution entière. Pour chaque route r, le centre de l'agrégat est défini comme le centre de gravité des clients qui constituent r, et les degrés

d'appartenance sont calculés par rapport à ce centre. Par conséquent, pour une route donnée r, son centre étant fixe, les degrés d'appartenance se calculent en une et une seule itération. Pour plus de cohérence et de clarté, on parlera de degrés d'appartenance $M(r, n)$ d'un client n à une route r. Notons que par définition, rien n'empêche un client n appartenant à la route r_1 dans la solution courante d'avoir un degré d'appartenance plus élevé pour une autre route r_2 (soit $M(r_1, n) < M(r_2, n)$). Cette mesure peut donc être vue comme un potentiel d'appartenance.

Voici un exemple sur une instance de 300 clients et 4 routes. Dans le premier graphique (Figure 6.1), nous pouvons identifier d'une part la distribution géographique des clients (cercles noirs), et d'autre part l'enveloppe convexe de chacune des routes et leur centre de gravité, matérialisé par les losanges colorés. Le losange noir représente la localisation du dépôt. Pour alléger la figure, les arcs entre clients des routes n'ont pas été représentés.

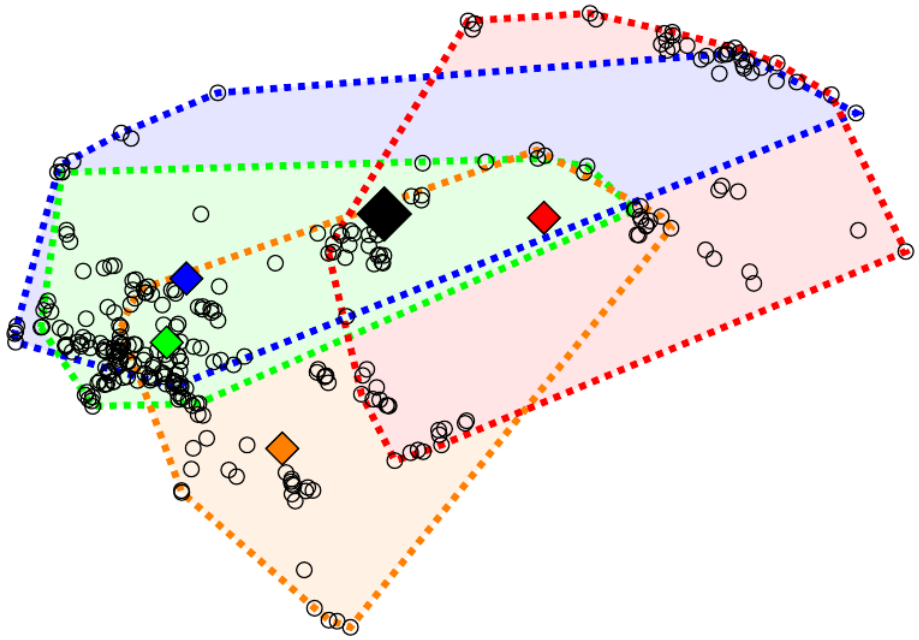


FIGURE 6.1 Exemple d'instance (avec les enveloppes convexes des routes)

Dans la figure suivante (Figure 6.2), nous avons représenté les degrés d'appartenance sous forme de carte thermique. Pour chaque petite zone de l'instance, découpée en carré, la couleur représente la moyenne des degrés d'appartenance des clients s'y trouvant. Les zones blanches correspondent aux zones dans lesquelles il n'y a aucun client. Pour chaque sous-figure, le centre pour lequel le degré d'appartenance est représenté est celui de plus grande taille (*e.g.*

le centre vert pour la sous-figure en haut à gauche).

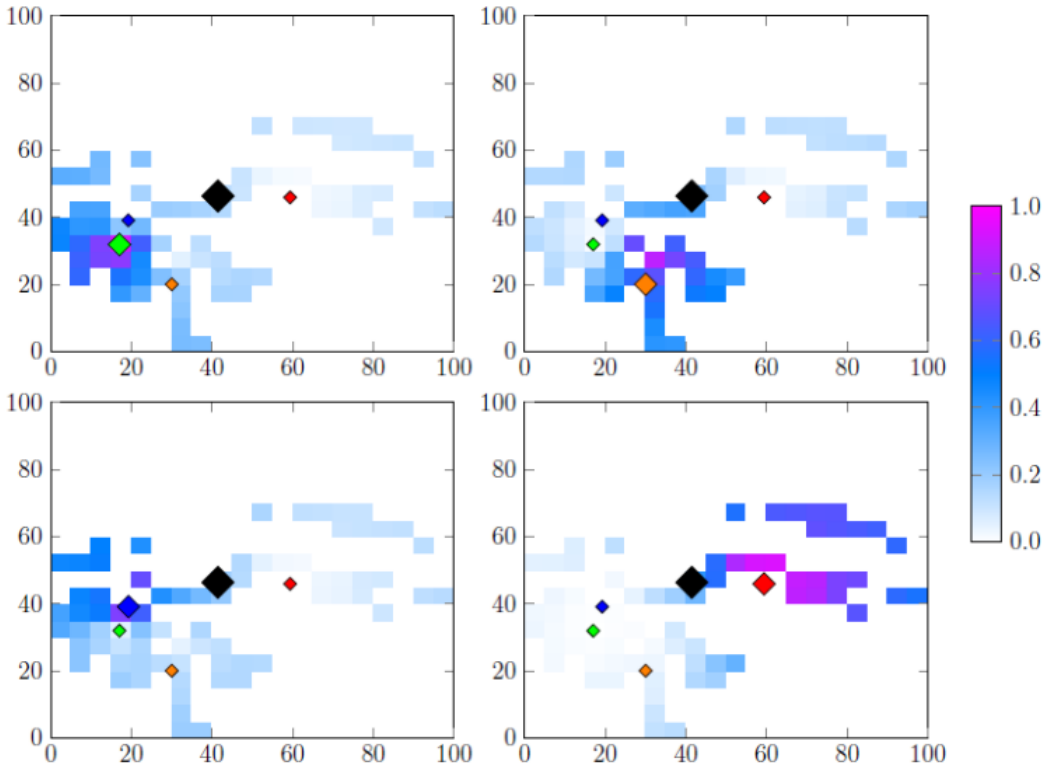


FIGURE 6.2 Degrés d'appartenance

On voit ici tout l'intérêt d'utiliser les degrés d'appartenance pour définir des voisinages plutôt que de simples seuils sur les distances. En effet, en regardant le dernier graphique, et les degrés d'appartenance relatifs à la route rouge, nous pouvons observer que les clients situés tout à droite de l'instance, bien que relativement loin du centre de la route, ont un degré d'appartenance élevé. C'est tout à fait logique, car il serait naturel de les assigner à la route rouge plutôt qu'à une autre, car les autres centres sont encore plus distants. Notons que ces degrés d'appartenance n'ont de sens et ne peuvent être calculés qu'avec des solutions entières.

6.2.3 Voisinages

Grâce aux agrégats de routes présentés précédemment, nous avons pu définir des voisinages dynamiques pour les routes. Ceux-ci ont un impact sur l'aspect combinatoire de la phase de recherche taboue utilisée dans la génération de colonnes heuristique. Soit $M(r)$ le vecteur des degrés d'appartenance de tous les clients de V relativement à la route r et V_r l'ensemble des clients appartenant à r. Définissons $V^C(r,p)$ l'ensemble des p clients n'appartenant pas à la

route r ayant les plus grandes valeurs dans $M(r)$. Ainsi, si $x \in V^C(r,p)$ (avec $|V^C(r,p)| = p$), nous avons $M(r,x) \geq M(r,x')$, $\forall x' \in V \setminus (V_r \cup V^C(r,p))$. Pour garder un maximum de contrôle sur la taille de chaque voisinage d'une route r, nous les définissons alors comme $\mathcal{N}_p(r) = V_r \cup V^C(p,r)$. Nous avons décidé de systématiquement inclure les clients de la route (V_r) dans le voisinage pour conserver une réalisabilité en tout temps. En effet, il se pourrait qu'un des clients ait un très faible degré d'appartenance à la route et ne soit pas sélectionné dans le voisinage considéré.

6.2.4 Sélection des routes

Si le concept d'agrégation de clients autour des routes se justifie aisément à partir d'une solution entière comme la solution courante au début d'une itération de LNS, notre algorithme a aussi dû être ajusté afin que son fonctionnement en cours de génération de colonnes, où de nouvelles routes apparaissent, continue à bénéficier de cette stratégie de réduction de l'espace des solutions.

Pour pouvoir générer de nouvelles colonnes dans un espace de solutions réduit grâce à nos voisinages précédemment définis, nous devons choisir judicieusement les colonnes sur lesquelles nous allons prodéder à de la recherche taboue. À chaque itération de génération de colonnes, la solution courante peut être fractionnaire. Or nous ne pouvons calculer de degrés d'appartenance que pour des solutions entières, et il en va par conséquent de même pour les voisinages tels que nous les avons définis. Originellement, les mouvements tabous étaient effectués à partir de colonnes en base (pas nécessairement entières). Afin de pouvoir considérer un maximum de colonnes avec des voisinages demeurant pertinents, nous proposons donc un mécanisme de transmission des voisinages par hérédité des routes. Pour une nouvelle colonne r' créée à partir de mouvements tabous sur une colonne r, nous définissons son hérédité avec l'attribut $Mere(r') = r$.

L'ensemble Ω^{it} contient les colonnes pouvant être modifiées au cours de cette itération de reconstruction. Il est initialisé avec les colonnes $\mathcal{M}$ de la solution courante (étape 1). Notons que par construction, $\Omega^{it} \subseteq \Omega^{dest}$. Nous commençons alors par initialiser l'attribut $Mere$ de chaque route r de $\mathcal{M}$ à elle-même. Après avoir résolu le PMR avec les colonnes dans le bassin Ω^{dest}, nous pouvons commencer la recherche taboue. Au lieu de pouvoir prendre n'importe quelle colonne de Ω^{dest} en base afin de lui appliquer des mouvements tabous, nous nous restreignons à celle de Ω^{it} (étape 8). Par conséquent, nous nous assurons d'avoir un voisinage bien défini pour ces routes (étape 9) et ainsi un espace de recherche réduit pour être plus efficace. Les routes r' nouvellement générées à partir de r héritent de la même mère que cette dernière (étape 11). De plus, elles sont ajoutées aux deux ensembles Ω^{dest} et Ω^{it}

Algorithme 6 Génération de colonnes (Routes mères/Routes filles)

Entrées : S : la solution entière courante ; $\mathcal{M}$: l'ensemble des routes de S, appelées routes mères

Ensembles : Ω^{dest} le sous-ensemble de colonnes compatibles avec la destruction dans le PMR ; Ω^{it} les colonnes modifiables de cette itération de TDR

1: $\Omega^{it} \leftarrow \mathcal{M}$
2: **for** $r \in \mathcal{M}$ **do**
3: $\quad Mere(r) = r$
4: **end for**
5: Résoudre PMR
6: **while** Conditions de poursuite de recherche de colonnes valides **do**
7: $\quad$ **while** Condition d'arrêt tabou non satisfaite **do**
8: $\quad\quad$ Choisir une route r en base dans Ω^{it}
9: $\quad\quad$ Chercher, par méthode taboue, dans le voisinage $\mathcal{N}_p(Mere(r))$, un ensemble R' de colonnes r' à coûts réduit négatif
10: $\quad\quad$ **for** $r' \in R'$ **do**
11: $\quad\quad\quad Mere(r') \leftarrow Mere(r)$
12: $\quad\quad\quad \Omega^{dest} = \Omega^{dest} \cup \{r'\}$ et $\Omega^{it} = \Omega^{it} \cup \{r'\}$
13: $\quad\quad$ **end for**
14: $\quad$ **end while**
15: $\quad$ Résoudre PMR
16: $\quad$ **if** Nouvelle solution entière **then**
17: $\quad\quad$ Mettre à jour les degrés d'appartenance et voisinages pour les colonnes en base
18: $\quad$ **end if**
19: **end while**

(étape 12). Après une nouvelle résolution du PMR (étape 15), parmi ces nouvelles colonnes, celles qui entrent en base deviennent aussi éligibles à être modifiées à leur tour, qu'elles aient des valeurs entières ou fractionnaires. Enfin, si une nouvelle solution entière est trouvée, les degrés d'appartenance et voisinages sont recalculés (étape 17) pour ces routes. La procédure sera détaillée plus précisément dans la section suivante.

6.2.5 Mise à jour des voisinages

La définition de nos voisinages étant uniquement basée sur des solutions entières, nous les mettons à jour à chaque nouvelle solution entière obtenue. Cela implique notamment de recalculer la position des centres des routes et les différentes matrices de degrés d'appartenance M. Ce processus nous permet de mieux suivre la solution courante au cours d'une même itération de reconstruction. Le centre des agrégats étant fixé au centre de gravité de chaque route, le calcul est rapide et efficace, même pour des instances de plus de 3000 clients et n'a donc pas d'impact sur le temps d'exécution de notre méthode. Nous pouvons ainsi nous

permettre de les recalculer à chaque nouvelle solution entière S'. Nous réinitialisons alors l'hérédité des routes r de S' ($Mere(r) = r$) et les voisinages utilisés jusque là sont conservés. De cette façon, chaque route de Ω^{it} est rattachée à un unique voisinage.

6.3 Partition des instances

Nos analyses initiales nous ont aussi amenés à d'autres observations. À cause de la grande taille des instances et des opérateurs de destruction, il existe de grandes zones géographiques sans destruction ou avec une trop faible densité de clients retirés pour pouvoir reconstruire de manière pertinente. En effet, il n'est pas envisageable de retirer trop de clients (en pratique plus de 500) car cela affecte grandement la performance de reconstruction. À chaque itération de LNS, l'essentiel de l'effort de recherche est donc localisé dans une zone relativement restreinte de l'instance, mais laisse les autres inchangées. La qualité des routes finales d'une région dépend donc de la fréquence à laquelle les opérateurs de destruction ont retiré des clients dans son voisinage géographique. Les opérateurs de destruction étant partiellement aléatoires, il est difficile d'avoir un contrôle total sur ceux-ci. Pour pallier ce problème, nous avons décidé de découper les grandes instances en plusieurs sous-instances avant d'utiliser, sur chacune d'entre elles et en parallèle, notre algorithme de base, pour un nombre d'itérations plus limité. L'objectif est d'intensifier les recherches dans toutes les zones géographiques de l'instance originale.

6.3.1 Génération des sous-instances

Pour créer des sous-instances, plusieurs critères sont à prendre en considération :

1. Les clients dans chacune des partitions doivent être choisis de manière à pouvoir former des routes compactes ;
2. Les sous-instances doivent être résolues rapidement, ce qui impose d'avoir déjà de bonnes solutions initiales pour chacune d'elles.

Pour répondre au premier critère, procéder à de l'agrégation de clients semble tout à fait pertinent. En revanche, pour ne pas avoir besoin de générer de nouvelles routes initiales et ainsi répondre au deuxième critère, l'agrégation de routes est plus appropriée. À partir de la solution courante S, composée des routes R^S, nous allons créer k sous-instances. Nous résolvons alors un problème de partition sur une instance auxiliaire composée de $|R^S|$ points, localisés aux centres de gravité de chacune des routes de S. Une fois le partitionnement $\{R^S_1, R^S_2, \ldots, R^S_k\}$ réalisé, chaque sous-instances I_l est générée à partir des clients des routes R^S_l, $\forall l \in \{1, \ldots, k\}$.

6.3.2 Résolution du problème de partition (PP)

De notre modélisation du PP va grandement dépendre la composition des sous-instances. En plus des contraintes classiques d'un PP, nous nous devons de faire attention à deux critères. Dans un premier temps, il serait préférable que les sous-instances générées soient de tailles homogènes. En effet, comme elles seront résolues de manière indépendante, mais en parallèle, des sous-instances de tailles similaires favoriseront des temps d'exécution homogènes. Par conséquent, les résolutions seront naturellement mieux synchronisées, bien que ce comportement ne soit pas garanti. Le deuxième critère concerne la qualité des partitions, notamment quand elles sont soumises à des contraintes d'équilibrage. Pour avoir des reconstructions pertinentes, mais aussi pour favoriser la création de routes compactes, il est préférable d'avoir des partitions elles-mêmes compactes, sans route isolée.

6.3.2.1 Agrégation avec *k-means*

Quand vient le moment de faire des agrégations, *k-means* revient sur le devant de la scène, de par sa mise en place facile. En revanche, il n'est pas possible de gérer la taille des partitions directement, ce qui, dans notre cas, s'avère problématique. Nous procédons alors en deux étapes. Dans un premier temps, nous utilisons *k-means* afin d'identifier les centres des k agrégats c_l ($l \in \{1, \ldots, k\}$). Par la suite, nous cherchons à agréger les routes autour de ces centres. Pour ce faire, nous résolvons un problème de partitionnement avec ces k centres ((6.1)-(6.4)), en cherchant à minimiser la distance au carré entre les centres des routes g_r et le centre c_l auquel elles sont effectivement affectées dans notre partitionnement ($\alpha_{lr} = 1$).

$$\min \sum_{r \in R^s} \sum_{l=1}^{k} \alpha_{lr} d(c_l, g_r)^2 \tag{6.1}$$

$$\text{subject to :} \quad \sum_{l=1}^{k} \alpha_{lr} = 1, \quad \forall r \in R^S \tag{6.2}$$

$$\alpha_{lr} \in \{0, 1\}, \quad \forall (l, r) \in \{1, \ldots, k\} \times R^s \tag{6.3}$$

$$\textit{Contraintes d'équilibrage} \tag{6.4}$$

Plus de détails sur les contraintes d'équilibrage (6.4) seront donnés dans la section 6.3.3. L'inconvénient majeur de cette approche vient de la localisation des centres des agrégats, principalement quand il y a des routes un peu isolées d'un foyer de haute densité.

6.3.2.2 Agrégation avec des degrés d'appartenance

Une seconde approche consiste à reprendre l'idée des agrégations floues pour extraire des degrés d'appartenance. Avec une telle approche, les chances d'avoir de nombreux agrégats isolés loin des zones de plus forte concentration sont plus faibles, comme nous le montre la figure 6.3 (3090 clients, 27 routes). Les cercles représentent les centres de gravité des 27 routes de R^S, les centres des agrégats donnés par l'algorithme de *k-means* sont représentés par les triangles bleus, alors que ceux donnés par les agrégations floues et les degrés d'appartenance sont les losanges rouges. Ces derniers centres s'obtiennent après avoir calculé les degrés d'appartenance par la méthode introduite par Bezdek *et al.* [62] dont le pseudo-code de l'algorithme 5 présente les grandes lignes. Abstraction faite de la route au sud-ouest qui est vraiment trop isolée du reste de l'instance, nous pouvons observer que les deux algorithmes gèrent différemment les agrégations.

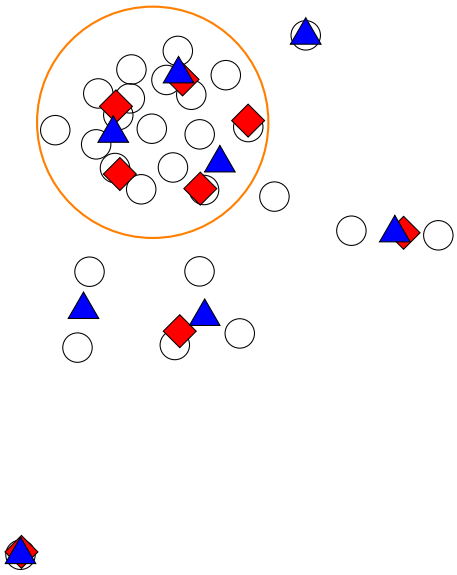


FIGURE 6.3 Centres des agrégats

Il est alors possible de résoudre un problème de partition similaire à celui présenté précédemment ((6.1)-(6.4)) en remplaçant les centres déterminés par *k-means* par ceux identifiés grâce aux degrés d'appartenance.

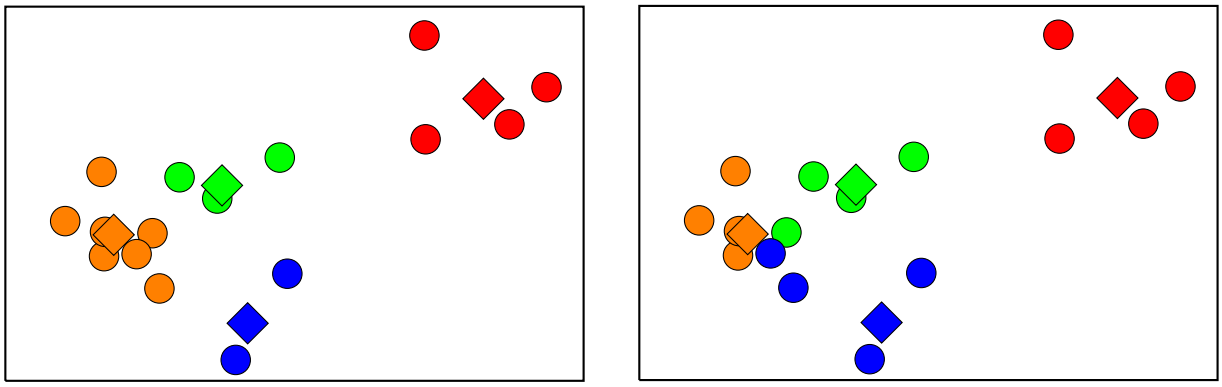
6.3.3 Options d'équilibrage

L'intérêt de ne pas utiliser directement une méthode d'agrégation comme "k-means" est de pouvoir garder le contrôle sur la taille des différents agrégats. Pour ceci, nous avons décidé d'ajouter au PP des contraintes d'équilibrage de la forme :

$$\lfloor |R^S|/k \rfloor - \tau \leq \sum_{r \in R^S} \alpha_{lr} \leq \lceil |R^S|/k \rceil + \tau, \quad \forall l \in \{1, \ldots, k\} \tag{6.5}$$

où τ est un paramètre de tolérance, et $\lfloor X \rfloor$ (resp. $\lceil X \rceil$) désigne le plus grand (resp. plus petit) nombre entier inférieur ou égal (resp. strictement supérieur) à X.

Dans la figure 6.4, nous pouvons voir le résultat d'un partitionnement sans contrainte (6.4a) et avec contraintes d'équilibrage (6.4b) pour une instance de 1555 clients, 16 véhicules et $k = 4$. Notons que sans équilibrage, la partition orange comporte 727 clients alors que la partition bleue en contient 184. Un tel cas de figure n'est pas avantageux pour deux raisons principales. La première est qu'en termes de temps de calcul, le risque d'une mauvaise synchronisation est relativement grand en raison de l'écart de taille des partitions. La deuxième concerne directement les performances. Avec une sous-instance comprenant seulement deux routes, les chances d'améliorations s'amenuisent, notamment par rapport à la mesure de compacité.



(a) Partition initiale - Sans équilibrage

(b) Partition initiale - Équilibrage des routes

FIGURE 6.4 L'importance de l'équilibrage

Nous avons aussi essayé d'équilibrer les partitions relativement au nombre de clients, mais nos résultats préliminaires ne nous ayant pas donné entière satisfaction, nous nous sommes concentrés sur l'équilibrage du nombre de routes.

6.3.4 Séquencement de partitions

La combinaison des différentes zones géographiques de l'instance se fait par l'intermédiaire d'un séquencement de partitions. En répétant ce schéma de partitions plusieurs fois, en changeant le nombre de partitions k, il est alors possible de recombiner de manière différente les

routes qui se trouvaient à la frontière d'une partition. Soit Seq la séquence de partitionnement définie par une suite d'entiers représentants k, par exemple $Seq = 8 - 6 - 5 - 4 - 3$. Dans cet exemple, l'instance est décomposée en 8 sous-instances avant de les résoudre puis de regrouper toutes les routes obtenues. La nouvelle solution globale courante est alors décomposée à son tour en 6 sous-instances, résolues puis groupées, et ainsi de suite. Notons que pour améliorer les performances du séquencement, il est judicieux de ne jamais avoir deux itérations consécutives de la séquence avec la même valeur. En effet, cela entrainera de manière quasi systématique le même partitionnement de sous-instances. Ceci est dû au fait que même si des échanges sont faits et que les centres des routes se déplacent d'une itération à l'autre, ces variations ne sont généralement pas suffisamment significatives pour modifier le partitionnement. Dans les sections suivantes, nous ferons référence à cette approche en séquencement de partitions par l'acronyme SDP.

6.4 Résultats

Nous avions à notre disposition deux instances industrielles avec respectivement 1555 et 3090 clients, que nous dénoterons I1555 et I3090 par la suite. Nous allons dans un premier temps présenter les résultats comparatifs entre notre solution et celle de notre partenaire obtenus sur ces instances complètes. Les deux méthodes ont le même point de départ, une solution initiale fournie elle aussi par notre partenaire. Dans un second temps, nous présenterons des résultats obtenus sur ces mêmes instances, mais où l'objectif est simplement d'améliorer la compacité de routes déjà optimisées, à la manière d'un post-traitement. Pour chaque jeu de paramètres, les résultats ont été reproduits trois fois avec des germes différents. Sauf indication contraire, les résultats présentés dans les tableaux sont les résultats moyens.

6.4.1 Méthode complète

Dans cette section, nous allons montrer l'impact de nos différentes techniques mises en place sur la qualité des résultats. Nous allons nous comparer à la solution finale générée par la version adaptée de GeoRoute. En effet, dans la pratique, le logiciel de Giro est capable de prendre en considération beaucoup de contraintes techniques que nous n'avons pas modélisées de notre côté. Celles-ci ont été temporairement désactivées sur GeoRoute afin que nous résolvions exactement le même problème et que nous puissions comparer les méthodes sur un pied d'égalité.

6.4.1.1 Comparaison des méthodes

Les Figures 6.5 et le Tableau 6.2 permettent une comparaison des différentes méthodes détaillées ci-dessous.

Méthode originale (Orig.)

La première méthode de résolution présentée correspond à la version originale de l'algorithme prenant en compte la compacité, similaire à celui présenté au Chapitre 5, avec 50 itérations de LNS.

Limitation des voisinages (Voisi.)

Les paramètres sont identiques à la méthode originale (Orig. avec 50 itérations de LNS), mais les voisinages considérés lors des mouvements tabous sont limités aux 250 meilleurs candidats selon leurs degrés d'appartenance ($\mathcal{N}_{250}(r)$) comme définis dans la Section 6.2.3.

Séquence de partitionnement (Seq1, Seq2, Seq3)

Nous avons par la suite testé le partitionnement d'instance. Chacune des partitions est alors résolue en parallèle pour 10 itérations de LNS, en conservant les voisinages de route $\mathcal{N}_{250}(r)$. Trois séquencements différents (adaptés à la taille des instances) ont été testés pour chacune des instances et sont définis dans le Tableau 6.1.

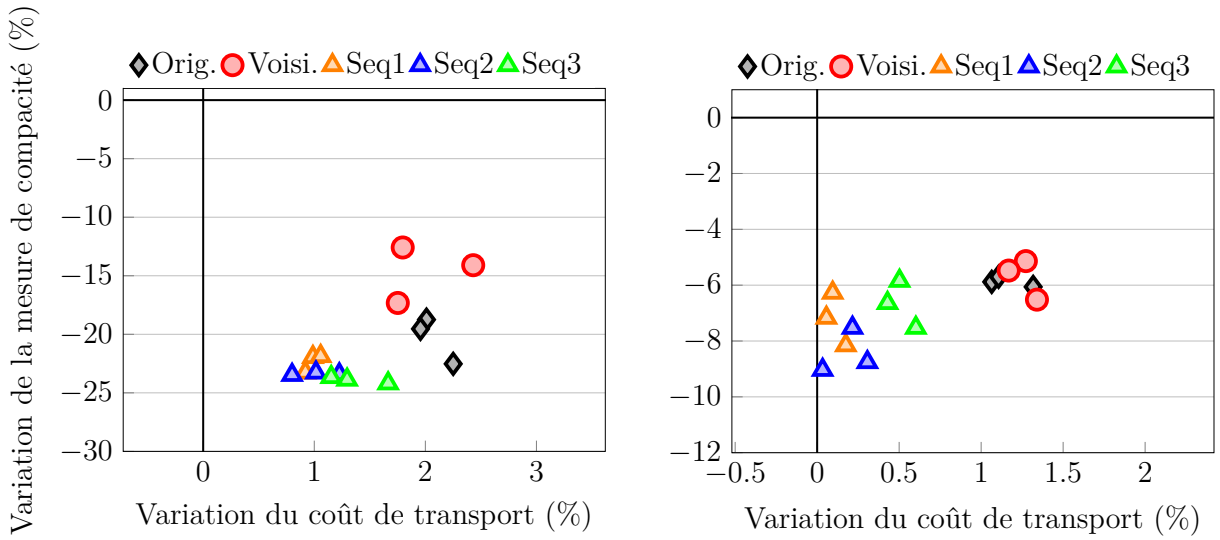
Tableau 6.1 Différents séquencements mis à l'essai

Nom	I1555	I3090
Seq1	4-3-2-4-3-2	8-7-6-5-4-3-2
Seq2	4-3-4-2-4-3	8-4-7-3-6-2-5-4
Seq3	3-4-3	8-5-3

Ces différents séquencements ont été choisis selon divers critères. Tout d'abord, nous voulions nous ramener, pour la toute première subdivision, à des tailles de sous-instances pour lesquelles nous savions que notre algorithme donnait des résultats pertinents. En utilisant 3 ou 4 subdivisons (resp. 8) pour 1555 clients (resp. 3090), les sous-instances générées contiennent généralement un nombre de clients de l'ordre de 400 ou 500, ce qui nous ramène à la taille des instances traitées dans le chapitre 5. Ensuite, au cours de nos phases de tests, nous avons pu observer que les séquences de type géométrique (par exemple 8-4-2-1) tendaient vers une séparation plus franche entre les groupes de routes que des séquences comprenant notamment des subdivisions impaires. De la longueur des séquencements dépend la qualité des solutions ainsi que le temps d'exécution, c'est pourquoi nous présentons des résultats pour des séquencements axés sur la qualité des solutions ($Seq1$, $Seq2$) ainsi que des séquencements courts pour les temps de résolution réduits ($Seq3$). Enfin, nous avons pu observer qu'une ultime

itération en considérant l'instance toute entière n'avait pas d'intérêt au niveau de la qualité des solutions obtenues et ne faisait que dégrader la performance en terme de temps de calcul.

Une étude comparative des performances de chaque méthode est représentée en Figure 6.5. Elle indique pour chacune des deux instances et pour chacune des solutions obtenues avec les différentes méthodes les déviations en % entre notre solution et celle fournie par la version adaptée de GeoRoute, par rapport aux critères de coût de transport (en abscisses) et de mesure de la compacité exacte (en ordonnées). Rappelons que dans cette version de GeoRoute, la compacité n'est nullement prise en considération. Nous pouvons par conséquent voir que toutes nos méthodes permettent d'obtenir un gain substantiel sur la mesure de la compacité, de 12 à 25 % (resp. 5 à 9 %) pour l'instance I1555 (I3090, respectivement). Les meilleurs sont d'ailleurs obtenus pour les méthodes SDP, tout comme l'impact sur le coût de transport. Ceux-ci varient entre 0.7 et 2.3%. Après présentation de nos résultats préliminaires à notre partenaire industriel, il est ressorti que l'augmentation du coût de transport pouvait être un réel frein à l'acceptation de solutions, même bien plus compactes. La tolérance vis-à-vis de cet impact économique reste au final à la discretion de l'utilisateur final, mais nous estimons qu'au-delà d'une hausse de 1% sur le coût de transport, les décideurs ont de plus fortes chances de refuser notre solution compacte.



(a) 1555 clients

(b) 3090 clients

FIGURE 6.5 Comparaison des différentes approches

En plus de la qualité des solutions, il est pertinent de comparer les temps d'exécution des différentes méthodes, présentées au tableau 6.2. Les temps de calcul annoncés pour la so-

lution de GeoRoute servent ici aussi de référence. La comparaison directe est avant tout donnée à titre indicatif, les calculs n'ayant pas été exécutés sur les mêmes machines. Cela permet tout de même de donner un ordre d'idée sur les potentielles améliorations que nous apportons. Nous porterons avant tout un regard critique entre la méthode originale (Orig.) et les autres. Comme nous pouvons le constater, la limitation des voisinages (Voisi.) permet bien de diminuer les temps de calcul par rapport à la méthode originale, bien que la qualité des solutions soit légèrement moins bonne (voir Figure 6.5). Couplée à la méthode SDP, elle permet aussi de gagner significativement du temps de calcul effectif, en plus d'obtenir des solutions de meilleure qualité. Il est évident qu'en parallélisant les exécutions, nous décuplons la puissance de calcul, mais n'allant pas au-dessus de 4 à 8 partitions, nous jugeons que c'est parfaitement en adéquation avec les technologies actuelles. Notons d'ailleurs que l'implémentation de la parallélisation des tâches n'a pas bénéficié d'un développement optimal et que des gains non négligeables pourraient être faits de ce côté. De manière logique, le temps d'exécution de la méthode SDP dépend largement de la taille du séquencement. C'est pourquoi Seq3 est si rapide comparativement aux autres, car il ne comporte que trois itérations de partitions.

Tableau 6.2 Temps de calcul moyens sur les instances industrielles

	I1555		I3090	
Variante	T (s)	Δ T (%)	T (s)	Δ T (%)
GeoRoute	3406	–	4569	–
Orig.	3706	8.8	3643	-20.3
Voisi.	2635	-22.6	2942	-35.6
Seq1	1730	-49.2	2591	-43.3
Seq2	1445	-57.6	3003	-34.3
Seq3	856	-74.9	1199	-73.7

6.4.1.2 Autres paramètres

Afin de justifier le choix de certaines valeurs de paramètres, nous présentons dans le tableau (6.3) quelques résultats avec des configurations légèrement différentes. Les résultats reflètent la moyenne des trois séquencements présentés Tableau 6.1, la ligne "Sect. 6.4.1.1" étant la moyenne de $Seq1$, $Seq2$, $Seq3$ présentés à la Figure 6.5. Par conséquent, les résultats représentent la moyenne de 9 exécutions, soit 3 germes différents pour chacune des 3 séquences. Pour chacune des alternatives, nous analysons alors la variation en terme de coût de transport (ΔR), de coût de compacité (ΔC) et de temps d'exécution (ΔT).

La première alternative proposée consiste en l'annulation de l'utilisation de voisinages lors des résolutions avec séquencement (Sans-$\mathcal{N}$). En effet, il pourrait paraître discutable de continuer

à limiter les voisinages considérés lorsque nous résolvons des sous-instances de taille réduite. Si les résultats sont relativement similaires, nous pouvons observer le fort impact sur le temps de calcul (près de 30% de sauvé en moins).

La seconde alternative est relative au PP, c'est-à-dire à notre façon de découper les instances. Si pour les résultats présentés précédemment (Section 6.4.1.1) les contraintes d'équilibrage n'avaient aucune tolérance, nous avons utilisé ici $\tau = 1$. Encore une fois, l'impact sur la qualité des solutions est relativement mineur, la plus grosse différence se faisant sur le temps de calcul. Comme expliqué précédemment, une plus grande tolérance augmente la différence de taille des sous-instances et nuit à la bonne synchronisation des résolutions.

Enfin, nous avons voulu voir les résultats que nous pouvions obtenir en augmentant le poids accordé à la compacité λ, en le multipliant par un facteur 4. Comme nous avions pu l'observer dans le chapitre précédent (Chapitre 5), ce genre d'ajustement conduit ici aussi à l'obtention de routes plus compactes, mais aussi légèrement plus longues.

NB : si pour $\lambda \times 4$, les gains sont très homogènes pour l'instance I1555, nous pouvons noter que pour I3090, il existe parmi les 9 exécutions une solution améliorant la compacité de 13.4%.

Tableau 6.3 Variations moyennes pour les paramètres alternatifs

	I1555			I3090		
Alternative	ΔR (%)	ΔC (%)	ΔT (%)	ΔR (%)	ΔC (%)	ΔT (%)
Sect. 6.4.1.1	1.12	-23.22	-60.55	0.27	-7.44	-50.43
Sans-$\mathcal{N}$	0.28	-22.71	-30.31	0.25	-7.54	-22.41
$\tau = 1$	0.91	-23.65	-47.17	0.23	-6.28	-29.48
$\lambda \times 4$	1.70	-24.72	-48.90	0.99	-9.59	-42.77

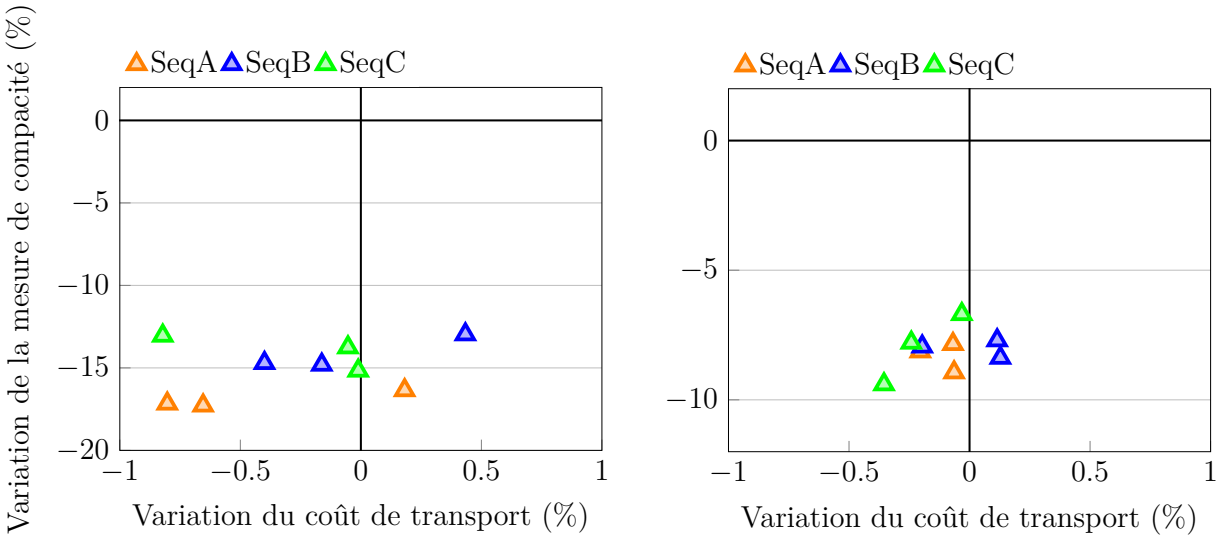
6.4.2 Post-traitement

Suite à nos premiers résultats, nous nous sommes aussi demandé s'il ne pouvait pas être pertinent d'utiliser notre méthode en guise de post-traitement, appliqué directement sur la solution finale fournie par Giro. En ajustant quelques paramètres pour rendre l'algorithme plus rapide (potentiellement au détriment de la qualité des solutions) et en choisissant des séquences courtes pour la SDP (voir tableau 6.4), nous avons obtenu les résultats présentés à la Figure 6.6.

Chacune des solutions obtenues permet un net gain relativement à la compacité des routes (de 13% à 17.3 % pour la première instance et de 6.7% à 9.4% pour la seconde), bien que celui-ci soit légèrement inférieur que dans la Section 6.4.1.1. En revanche, nous n'observons

TABLEAU 6.4 Différents séquencements pour le post-traitement

Nom	I1555	I3090
SeqA	3-4-3	8-5-3
SeqB	4-3-2	8-6-4
SeqC	4-2-4	8-7-6



(a) 1555 clients

(b) 3090 clients

FIGURE 6.6 Post-traitement

pas le même impact sur le coût de transport puisque pour chacune des deux instances, 7 des 9 exécutions ont même permis de réduire ce coût. De plus, avec les paramètres utilisés, les temps d'exécution moyens sont faibles (considérant la grande taille des problèmes) avec respectivement 621s. et 785s., soit moins de 15 minutes.

6.5 Conclusion

Nous avons mis en avant dans ce chapitre des approches permettant de traiter de très grandes instances pour le problème de routes compactes lors de livraison de colis dans le milieu postal. Nous avons su adapter la méthode mise en place dans le chapitre 5 pour la rendre plus performante lorsque la taille des instances s'accroit. Dans un premier temps, nous nous sommes attaqués au sous-problème de la génération de colonne heuristique en limitant l'espace de recherche taboue à des voisinages définis par rapport à une nouvelle notion de degrés d'appartenance à une route. Cet outil permet d'avoir des voisinages dynamiques qui s'adaptent

à chaque nouvelle solution entière trouvée au cours du processus. Cet ajout a permis de diminuer le temps de calcul d'environ 15% à 30% sans réel impact sur la qualité des solutions. Nous avons ensuite proposé un séquencement de partitions afin de décomposer le problème et ainsi gagner en efficacité. Combiné aux voisinages dynamiques, le SDP permet non seulement d'obtenir des solutions de meilleure qualité, mais aussi de diminuer encore une fois les temps de calcul, parfois drastiquement (-75%) dépendamment des stratégies employées. Enfin, nous avons cherché à évaluer la pertinence de notre méthode en cas d'utilisation en post-traitement sur des routes déjà optimisées. Avec de petites séquences de SDP, nous avons pu obtenir des résultats encourageants vis-à-vis de la compacité en moins de 15 minutes de temps de calcul, le tout sans dégrader la qualité des solutions en termes de coûts de transport.

CHAPITRE 7 DISCUSSION GÉNÉRALE

Dans cette thèse nous avons étudié l'aspect spatio-temporel des routes du problème de livraison de colis dans un contexte postal. L'application à laquelle nous nous sommes intéressés possède quelques caractéristiques qui la rendent originale, notamment en raison de la taille des instances rencontrées, de la densité des fenêtres de temps ou encore de la non prise en compte de la capacité des véhicules. Dans ce chapitre, nous allons revenir sur les différentes méthodes mises en place au cours de cette thèse afin de mesurer leur impact, leurs limites ainsi que les éventuelles améliorations envisageables.

7.1 Synthèse des travaux

Au cours de cette thèse, nous avons développé des méthodes pour répondre à des problèmes de plus en plus proches des applications réelles, en ne considérant d'abord qu'un seul véhicule, puis plusieurs avant d'augmenter considérablement la taille des instances traitées.

Dans le chapitre 4, nous avons tout d'abord voulu montrer l'impact économique de considérer la durée des routes dans la fonction objectif des problèmes de livraison comportant des fenêtres de temps. En résolvant des TSPTW incluant à la fois le temps de parcours et la durée des routes dans la fonction objectif, nous avons pu montrer que la fonction objectif visant à minimiser les distances parcourues uniquement, utilisée de manière quasi-systématique dans ce genre de problème, n'était pas forcément la plus adaptée. Pour analyser cet impact économique sur les instances académiques, nous avons proposé un modèle de programmation par contraintes avec une fonction objectif pondérée (WOF). Avec les coûts horaires considérés, nous avons pu montrer que résoudre ce problème avec une approche multi-objectif permettait de réduire en moyenne de 2.5% le coût de la solution finale par rapport à une approche ne visant qu'à minimiser le temps de parcours. Afin de pouvoir traiter des problèmes de grands territoires présentés par notre partenaire Giro, comportant très peu de fenêtres de temps, nous avons aussi mis en place un mécanisme d'agrégation/désagrégation portant sur les clients sans fenêtre de temps. Lors de la désagrégation, nous nous sommes aussi appuyés sur un modèle MIP, basé sur une discrétisation de l'horizon temporel en seaux de temps, afin de minimiser au mieux le temps de parcours au cours de cette reconstruction. Cette méthodologie d'agrégation s'est montrée très performante pour résoudre les problèmes industriels à faible densité de fenêtre de temps comprenant jusqu'à 475 clients. Le gain par rapport à la méthode WOF est de 1.6% en moyenne en se limitant à un temps d'exécution de 90 secondes.

Dans le chapitre 5, nous nous sommes intéressés au problème de livraison de colis postaux de manière plus générale en traitant non plus un seul véhicule mais une flotte complète, dans un problème de VRPTW. Dans cet article, l'accent a été mis sur la mesure de compacité. L'objectif est d'obtenir les routes les plus compactes possible afin de faciliter à la fois le travail des livreurs et celui des personnes en charge de l'affectation des routes au sein de la compagnie postale. Pour ce faire, nous avons commencé par définir la compacité géographique d'une route comme étant l'aire de l'enveloppe convexe des clients qui la composent. Afin de simplifier son calcul dans notre méthode de résolution, basée sur un algorithme LNS dont la phase de reconstruction s'appuie sur une méthode de génération de colonnes heuristique utilisant la recherche taboue, nous avons proposé deux approximations à cette mesure de compacité considérée comme exacte. La première repose sur des rectangles circonscrivant les clients de la route, la deuxième sur un disque dont le diamètre correspond à la distance séparant les deux clients les plus éloignés de la route. Nous avons pu mettre en avant, dans un contexte académique avec de nombreuses fenêtres de temps, que temps de parcours et compacité constituaient souvent des objectifs antagonistes. En effet, nos résultats ont montré qu'un gain considérable pouvait être fait sur la compacité des routes, entre 30% à 40% en moyenne, mais que ce gain ne se faisait qu'au coût d'un transport plus important, avec une augmentation de l'ordre de 10% à 15%. Avec des instances générées à partir de données industrielles, comprenant notamment une densité moindre de fenêtres de temps, nous avons pu observer que la compacité des routes pouvait être améliorée d'environ 10% par rapport à la méthode originale sur laquelle nous nous sommes appuyés. De plus, il semblerait même que la compacité puisse avoir un rôle de guide dans notre méthode de résolution, étant donné que le coût de transport a lui aussi pu être amélioré par rapport à la méthode originale pour le même nombre d'itérations. Enfin, dans un contexte ne considérant que des temps de parcours et non pas des positions géographiques, nous avons pu voir que la mesure de compacité avec le disque pouvait tout à fait s'adapter à cet autre type de données, générant d'ailleurs des résultats tout aussi satisfaisants relativement à la compacité géographique.

Dans le chapitre 6, nous avons cherché à aborder ce même problème sur des instances de très grande taille (1500 et 3000 clients). À partir de la méthodologie appliquée au chapitre 5, notre objectif a été de rendre la résolution plus rapide et efficace afin de limiter l'impact de la combinatoire exponentielle propre à ces instances comptant parfois plus de 3000 clients. Nous nous sommes tout d'abord appuyés sur des voisinages basés sur une notion de degré d'appartenance entre les clients et les routes. Ces voisinages permettent de rendre la méthode de reconstruction par génération de colonnes heuristique plus efficace en se focalisant sur les clients prometteurs lors des phases de recherche par méthode taboue. En plus de cela, nous avons pu observer que la taille de l'instance impliquait naturellement que de grandes régions

de l'instance pouvaient rester intactes pendant de nombreuses itérations car les opérateurs de destruction ne s'y appliquaient pas. Nous sommes donc revenus à une approche pouvant sembler similaire à une forme de "*cluster-first, route-second*" où nous avons décidé de diviser l'instance initiale de grande taille en plusieurs sous-instances que nous pouvons résoudre en parallèle. En changeant successivement la taille de ces subdivisions pour limiter les effets de bord, nous avons obtenu des solutions dont les routes sont plus compactes que celles proposées par la version de GeoRoute adaptée à laquelle nous nous sommes comparés. Il faut noter que la compacité ne fait pas encore partie des critères pris en compte par la solution logicielle de Giro et qu'il est donc naturel que nos solutions soient plus compactes, mais l'impact sur les coûts de transport est très faible (entre 0% et 1.7% pour le cas le plus défavorable) alors que le gain en compacité varie de 6% à 24%. Le tout avec un temps d'exécution limité considérant la taille des instances. De plus, dans une optique de post-optimisation, nous avons pu obtenir des résultats très encourageants améliorant encore une fois la compacité (de 6% à 17%) sans dégrader les coûts de transport voire parfois en les améliorant, en moins de 15 minutes d'exécution.

7.2 Limitations de la solution proposée et améliorations futures

Au cours de cette thèse, nous avons parfois dû faire des choix méthodologiques ou des hypothèses simplificatrices afin de recadrer notre sujet d'étude.

Sur l'ensemble des projets, afin de représenter au mieux les réalités du trafic urbain, il pourrait être pertinent d'incorporer un aspect dynamique au temps de parcours afin de dissocier, *a minima*, les heures de pointe des heures de trafic régulières. D'un point de vue pratique, prendre en considération la probabilité de présence des clients au moment de la livraison pourrait avoir un impact considérable. Nous avons tous pu expérimenter au cours des dernières années que les différentes compagnies postales et/ou les différents livreurs ne gèrent pas de la même façon les livraisons chez les clients absents. Dans certains cas, le colis est déposé sur le palier, dans d'autres, un avis de passage est laissé et le colis est soit retourné à un comptoir de remises avoisinant (bureau de poste, pharmacie,...), soit au dépôt de la compagnie. Intégrer cette notion d'absence du client et de passage au comptoir de remises serait une suite logique à notre projet. De plus, l'appel à de tels recours s'inscrit parfaitement dans la recherche de routes compactes puisque les détours générés n'en seraient que plus faibles.

Dans le chapitre 4, nous avons testé notre méthode d'agrégation/désagrégation de clients sur des instances industrielles pour lesquelles la densité de clients avec fenêtres de temps était extrêmement basse. Il serait pertinent d'évaluer cette méthodologie avec des densités se rapprochant de celles étudiées dans les projets suivants, *i.e.* de l'ordre de 20% pour voir

si elle continue à performer. Il est d'ailleurs possible que le MIP basé sur les seaux de temps ne soit pas le plus performant pour compléter la phase de désagrégation, mais comme nous l'avions implémenté pour des tests préliminaires et qu'il performait tout de même de manière satisfaisante dans notre contexte, nous avons continué à l'utiliser.

Dans le chapitre 5, nous nous sommes principalement focalisés sur la notion de compacité géographique, en prenant pour acquis que nous connaissions les positions exactes de chacun des clients (latitude/longitude généralement). En pratique, cette information est généralement accessible, mais si seule une matrice de distances nous est fournie, alors seule l'approximation par diamètre est envisageable et toute comparaison avec la notion de compacité exacte telle que nous l'avons définie devient impossible. Il serait donc très pertinent de définir de nouvelles mesures de compacité basées uniquement sur ces matrices de distances par réseaux routiers (et non euclidiennes) ou par temps de parcours puisque ces dernières ont beaucoup de sens dans une application réelle. Nous pouvons noter que dans ce projet, comme dans le suivant, nous ne faisons que contrôler la durée des routes afin de s'assurer qu'elle ne dépasse pas la limite fixée par les paramètres. Elle ne fait donc pas partie de la fonction objectif. De manière quasi-systématique, le temps d'attente réel est nul. Une amélioration de la durée de la route passe donc inexorablement par une diminution du temps de parcours effectué le long de la route.

Dans le dernier chapitre, les résolutions des sous-instances sont limitées uniquement par un nombre d'itérations de LNS chacune. Nous n'avons pas mis en place de méthode plus avancée pour améliorer la synchronisation des résolutions, si ce n'est d'homogénéiser les sous-instances par leur nombre de routes, ce qui a nécessairement un impact sur les performances. Nous pourrions choisir de nous synchroniser sur la sous-instance dont la résolution est la plus longue, ce qui améliorerait potentiellement la qualité des solutions. Sinon, nous pourrions chercher à résoudre le problème encore plus rapidement en écourtant les résolutions après le retour de la moitié des solutions issues de la décomposition. Relativement à la parallélisation, une implémentation plus optimisée pourrait, elle aussi, permettre d'économiser un peu de temps de calcul.

CHAPITRE 8 CONCLUSION ET RECOMMANDATIONS

Cette thèse nous a permis d'étudier le problème de tournées de véhicules avec fenêtres de temps dans un contexte particulier qu'est le milieu postal. En plus des caractéristiques propres à ce secteur d'activités, aussi bien la taille des instances industrielles que la faible densité de fenêtres de temps ou l'aspect non contraignant de la capacité des véhicules, nous nous sommes focalisés sur les caractéristiques spatio-temporelles des routes. À travers la durée de celles-ci, facteur à l'impact économique fort comme nous avons pu le constater dans le premier article, ou de leur compacité comme nous l'avons étudié plus profondément dans les chapitres suivants, nous avons pu mettre en lumière plusieurs cas de figure où la simple minimisation des temps de parcours pouvait donner des solutions non-satisfaisantes voire sous-optimale dans un contexte pratique industriel.

Nous avons fait appel au cours de ces travaux à de nombreuses méthodes de résolutions différentes, notamment programmation par contraintes ou en nombre entiers, LNS, génération de colonnes, recherche taboue, agrégations, ... Nous les avons par la suite combinées pour créer des heuristiques nous permettant d'obtenir des solutions de qualité avec des temps de calculs compétitifs. En effet, c'est aussi un des défis propres à ce contexte de travail très dynamique, où les clients à visiter peuvent changer drastiquement d'un jour à l'autre.

Les différents travaux réalisés ont pu être présentés lors de diverses conférences au cours de cette thèse et nous avons pu constater un intérêt notable à la notion de compacité notamment d'acteurs industriels de manière générale, sans restriction à l'activité postale. Comme nous avons pu l'analyser, l'utilisation de mesures de compacité a un effet bien plus bénéfique sur les instances industrielles qu'académiques qui, si elles constituent parfois de beaux défis en termes de complexité de résolution, ne représentent pas forcément une activité industrielle conventionnelle. Ceci est encourageant quant à la mise en application de telles méthodologies dans des applications pratiques où les critères de qualité modélisés s'avèrent de plus en plus divers et multiples.

RÉFÉRENCES

[1] "Postes Canada à l'ère du numérique : document de travail," Gouvernement du Canada, 2016. [En ligne]. Disponible : https://www.tpsgc-pwgsc.gc.ca/examendepostescanada-canadapostreview/rapport-report/consult-fra.html

[2] "Retour sur l'année 2019," Postes Canada, 2020. [En ligne]. Disponible : https://www.canadapost.ca/scp/fr/notre-entreprise/a-notre-sujet/rapports-financiers/rapport-annuel-2019/apercu-2019.page

[3] "Les marchés du courrier, du colis et des activités connexes en France - année 2019," Autorité de régulation des communications électroniques, des postes et de la distribution de la presse (ARCEP), 2020. [En ligne]. Disponible : https://www.arcep.fr/cartes-et-donnees/nos-publications-chiffrees/observatoire-courrier-colis/lmarches-courrier-colis-activites-connexes-france-2019.html

[4] "Une année record pour colissimo ?" Stratégies Logistique, 2020. [En ligne]. Disponible : https://strategieslogistique.com/Une-annee-record-pour-Colissimo,10506

[5] "Livraisons de colis : occupé jusqu'en 2021, selon la direction de Purolator," Radio-Canada, 2020. [En ligne]. Disponible : https://ici.radio-canada.ca/nouvelle/1755073/purolator-volume-colis-hausse-livraison-fetes

[6] A. Bretin, G. Desaulniers et L.-M. Rousseau, "The traveling salesman problem with time windows in postal services," *Journal of the Operational Research Society*, vol. 72, n°. 2, p. 383–397, 2021.

[7] G. B. Dantzig et J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, n°. 1, p. 80–91, 1959.

[8] P. Toth et D. Vigo, *Vehicle routing : problems, methods, and applications*. SIAM, 2014.

[9] K. Braekers, K. Ramaekers et I. Van Nieuwenhuyse, "The vehicle routing problem : State of the art classification and review," *Computers & Industrial Engineering*, vol. 99, p. 300–313, 2016.

[10] N. Jozefowiez, F. Semet et E.-G. Talbi, "Multi-objective vehicle routing problems," *European Journal of Operational Research*, vol. 189, n°. 2, p. 293–309, 2008.

[11] J. Branke, K. Deb, K. Miettinen et R. Slowiński, *Multiobjective optimization : Interactive and evolutionary approaches*. Springer Science & Business Media, 2008, vol. 5252.

[12] N. Jozefowiez, F. Semet et E.-G. Talbi, "An evolutionary algorithm for the vehicle routing problem with route balancing," *European Journal of Operational Research*, vol. 195, n°. 3, p. 761–769, 2009.

[13] T. Bektaş et G. Laporte, "The pollution-routing problem," *Transportation Research Part B : Methodological*, vol. 45, n°. 8, p. 1232–1250, 2011.

[14] V. Pillac, M. Gendreau, C. Guéret et A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, n°. 1, p. 1–11, 2013.

[15] D. Bertsimas, "Probabilistic combinatorial optimization problems," Thèse de doctorat, Massachusetts Institute of Technology, 1988.

[16] D. J. Bertsimas, "A vehicle routing problem with stochastic demand," *Operations Research*, vol. 40, n°. 3, p. 574–585, 1992.

[17] G. Laporte, F. Louveaux et H. Mercure, "The vehicle routing problem with stochastic travel times," *Transportation science*, vol. 26, n°. 3, p. 161–170, 1992.

[18] P. Augerat, D. Naddef, J. Belenguer, E. Benavent, A. Corberan et G. Rinaldi, "Computational results with a branch and cut code for the capacitated vehicle routing problem," 1995, https ://www.osti.gov/etdeweb/biblio/289002.

[19] M. Desrochers, J. Desrosiers et M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Operations research*, vol. 40, n°. 2, p. 342–354, 1992.

[20] A. Langevin, M. Desrochers, J. Desrosiers, S. Gélinas et F. Soumis, "A two-commodity flow formulation for the traveling salesman and the makespan problems with time windows," *Networks*, vol. 23, n°. 7, p. 631–640, 1993.

[21] G. Laporte, M. Desrochers et Y. Nobert, "Two exact algorithms for the distance-constrained vehicle routing problem," *Networks*, vol. 14, n°. 1, p. 161–172, 1984.

[22] M. Savelsbergh, "The Vehicle Routing Problem with Time Windows : minimizing route duration." *ORSA Journal on Computing*, vol. 4, n°. 2, p. 146–154, 1992.

[23] M. López-Ibáñez, C. Blum, J. W. Ohlmann et B. W. Thomas, "The travelling salesman problem with time windows : Adapting algorithms from travel-time to makespan optimization," *Applied Soft Computing*, vol. 13, n°. 9, p. 3806–3815, 2013.

[24] I. Kara et T. Derya, "Formulations for minimizing tour duration of the traveling salesman problem with time windows," *Procedia Economics and Finance*, vol. 26, p. 1026–1034, 2015.

[25] S. Belhaiza, P. Hansen et G. Laporte, "A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows," *Computers & Operations Research*, vol. 52, p. 269–281, 2014.

[26] F. Tricoire, M. Romauch, K. F. Doerner et R. F. Hartl, "Heuristics for the multi-period orienteering problem with multiple time windows," *Computers & Operations Research*, vol. 37, n°. 2, p. 351–367, 2010.

[27] C. Malandraki et M. S. Daskin, "Time dependent vehicle routing problems : Formulations, properties and heuristic algorithms," *Transportation science*, vol. 26, n°. 3, p. 185–200, 1992.

[28] J.-F. Cordeau, G. Laporte et A. Mercier, "A unified tabu search heuristic for vehicle routing problems with time windows," *Journal of the Operational research society*, vol. 52, n°. 8, p. 928–936, 2001.

[29] ——, "Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows," *Journal of the Operational Research Society*, vol. 55, n°. 5, p. 542–546, 2004.

[30] B.-I. Kim, S. Kim et S. Sahoo, "Waste collection vehicle routing problem with time windows," *Computers & Operations Research*, vol. 33, n°. 12, p. 3624–3642, 2006.

[31] R. Z. Ríos-Mercado et E. Fernández, "A reactive GRASP for a commercial territory design problem with multiple balancing requirements," *Computers & Operations Research*, vol. 36, n°. 3, p. 755–776, 2009.

[32] F. Arnold et K. Sörensen, "What makes a VRP solution good ? The generation of problem-specific knowledge for heuristics," *Computers & Operations Research*, vol. 106, p. 280–288, 2019.

[33] D. L. Horn, C. R. Hampton et A. J. Vandenberg, "Practical application of district compactness," *Political Geography*, vol. 12, n°. 2, p. 103–120, 1993.

[34] L. Torres et B. Suggs, "Smart locker system and method of parcel delivery," juill. 2 2015, U.S. Patent App. 14/582,088.

[35] D. Dumez, F. Lehuédé et O. Péton, "A large neighborhood search approach to the vehicle routing problem with delivery options," *Transportation Research Part B : Methodological*, vol. 144, p. 103–132, 2021.

[36] C. C. Murray et R. Raj, "The multiple flying sidekicks traveling salesman problem : Parcel delivery with multiple drones," *Transportation Research Part C : Emerging Technologies*, vol. 110, p. 368–398, 2020.

[37] N. Agatz, P. Bouman et M. Schmidt, "Optimization approaches for the traveling salesman problem with drone," *Transportation Science*, vol. 52, n°. 4, p. 965–981, 2018.

[38] L. Costa, C. Contardo et G. Desaulniers, "Exact branch-price-and-cut algorithms for vehicle routing," *Transportation Science*, vol. 53, n°. 4, p. 946–985, 2019.

[39] M. Desrochers et F. Soumis, “A generalized permanent labelling algorithm for the shortest path problem with time windows,” *INFOR : Information Systems and Operational Research*, vol. 26, n°. 3, p. 191–212, 1988.

[40] G. Righini et M. Salani, “Symmetry helps : Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints,” *Discrete Optimization*, vol. 3, n°. 3, p. 255–273, 2006.

[41] D. Feillet, P. Dejax, M. Gendreau et C. Gueguen, “An exact algorithm for the elementary shortest path problem with resource constraints : Application to some vehicle routing problems,” *Networks*, vol. 44, n°. 3, p. 216–229, 2004.

[42] G. Desaulniers, F. Lessard et A. Hadjar, “Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows,” *Transportation Science*, vol. 42, n°. 3, p. 387–404, 2008.

[43] R. Baldacci, A. Mingozzi et R. Roberti, “New route relaxation and pricing strategies for the vehicle routing problem,” *Operations research*, vol. 59, n°. 5, p. 1269–1283, 2011.

[44] F. Glover, “Tabu search—part i,” *ORSA Journal on computing*, vol. 1, n°. 3, p. 190–206, 1989.

[45] ——, “Tabu search—part ii,” *ORSA Journal on computing*, vol. 2, n°. 1, p. 4–32, 1990.

[46] É. Taillard, P. Badeau, M. Gendreau, F. Guertin et J.-Y. Potvin, “A tabu search heuristic for the vehicle routing problem with soft time windows,” *Transportation science*, vol. 31, n°. 2, p. 170–186, 1997.

[47] Z. Fu, R. Eglese et L. Y. Li, “A unified tabu search algorithm for vehicle routing problems with soft time windows,” *Journal of the Operational Research Society*, vol. 59, n°. 5, p. 663–673, 2008.

[48] C. Archetti, M. G. Speranza et A. Hertz, “A tabu search algorithm for the split delivery vehicle routing problem,” *Transportation science*, vol. 40, n°. 1, p. 64–73, 2006.

[49] D. Pisinger et S. Ropke, “Large neighborhood search,” dans *Handbook of metaheuristics*. Springer, 2010, p. 399–419.

[50] E. Prescott-Gagnon, G. Desaulniers et L.-M. Rousseau, “A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows,” *Networks*, vol. 54, n°. 4, p. 190–204, 2009.

[51] D. Pisinger et S. Ropke, “A general heuristic for vehicle routing problems,” *Computers & Operations Research*, vol. 34, n°. 8, p. 2403–2435, 2007.

[52] N. Mladenović et P. Hansen, “Variable neighborhood search,” *Computers & operations research*, vol. 24, n°. 11, p. 1097–1100, 1997.

[53] M. Polacek, R. F. Hartl, K. Doerner et M. Reimann, "A variable neighborhood search for the multi depot vehicle routing problem with time windows," *Journal of heuristics*, vol. 10, n°. 6, p. 613–627, 2004.

[54] B. Yu, Z.-Z. Yang et B. Yao, "An improved ant colony optimization for vehicle routing problem," *European Journal of Operational Research*, vol. 196, n°. 1, p. 171–176, 2009.

[55] P. P. Repoussis, C. D. Tarantilis et G. Ioannou, "Arc-guided evolutionary algorithm for the vehicle routing problem with time windows," *IEEE Transactions on Evolutionary Computation*, vol. 13, n°. 3, p. 624–647, 2009.

[56] T. Vidal, T. G. Crainic, M. Gendreau et C. Prins, "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows," *Computers & Operations Research*, vol. 40, n°. 1, p. 475–489, 2013.

[57] Y. Nagata, O. Bräysy et W. Dullaert, "A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows," *Computers & Operations Research*, vol. 37, n°. 4, p. 724–737, 2010.

[58] A. K. Jain, M. N. Murty et P. J. Flynn, "Data clustering : a review," *ACM computing surveys (CSUR)*, vol. 31, n°. 3, p. 264–323, 1999.

[59] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," dans *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, n°. 14. Oakland, CA, USA, 1967, p. 281–297.

[60] T. M. Kodinariya et P. R. Makwana, "Review on determining number of cluster in k-means clustering," *International Journal*, vol. 1, n°. 6, p. 90–95, 2013.

[61] D. Arthur et S. Vassilvitskii, "k-means++ : The advantages of careful seeding," Stanford, Rapport technique, 2006.

[62] J. C. Bezdek, R. Ehrlich et W. Full, "Fcm : The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, n°. 2-3, p. 191–203, 1984.

[63] M. Savelsbergh, "Local search in routing problems with time windows," *Annals of Operations Research*, vol. 4, n°. 1, p. 285–305, 1985.

[64] N. Christofides, A. Mingozzi et P. Toth, "State-space relaxation procedures for the computation of bounds to routing problems," *Networks*, vol. 11, n°. 2, p. 145–164, 1981.

[65] N. Boland, M. Hewitt, D. M. Vu et M. Savelsbergh, *Solving the Traveling Salesman Problem with Time Windows Through Dynamically Generated Time-Expanded Networks*. Cham : Springer International Publishing, 2017, p. 254–262.

[66] R. Baldacci, A. Mingozzi et R. Roberti, "New State-Space Relaxations for Solving the Traveling Salesman Problem with Time Windows." *INFORMS Journal on Computing*, vol. 24, n°. 3, p. 356–371, 2012.

[67] M. Gendreau, A. Hertz, G. Laporte et M. Stan, “A generalized insertion heuristic for the traveling salesman problem with time windows,” *Operations Research*, vol. 46, n°. 3, p. 330–335, mar 1998.

[68] M. Gendreau, A. Hertz et G. Laporte, “New insertion and postoptimization procedures for the traveling salesman problem,” *Operations Research*, vol. 40, n°. 6, p. 1086–1094, 1992.

[69] R. W. Calvo, “A new heuristic for the traveling salesman problem with time windows,” *Transportation Science*, vol. 34, n°. 1, p. 113–124, 2000.

[70] J. W. Ohlmann et B. W. Thomas, “A compressed-annealing heuristic for the traveling salesman problem with time windows,” *INFORMS Journal on Computing*, vol. 19, n°. 1, p. 80–90, 2007.

[71] M. López-Ibáñez et C. Blum, “Beam-aco for the travelling salesman problem with time windows,” *Computers & Operations Research*, vol. 37, n°. 9, p. 1570 – 1583, 2010.

[72] R. Ferreira da Silva et S. Urrutia, “A general vns heuristic for the traveling salesman problem with time windows,” *Discrete Optimization*, vol. 7, p. 203–211, 2010.

[73] M. Savelsbergh, “An efficient implementation of local search algorithms for constrained routing problems,” *European Journal of Operational Research*, vol. 47, n°. 1, p. 75 – 85, 1990.

[74] N. Ascheuer, M. Fischetti et M. Grötschel, “Solving the asymmetric travelling salesman problem with time windows by branch-and-cut,” *Mathematical Programming*, vol. 90, n°. 3, p. 475–506, May 2001.

[75] S. Dash, O. G¨unl¨uk, A. Lodi et A. Tramontani, “A time bucket formulation for the traveling salesman problem with time windows,” *INFORMS Journal on Computing*, vol. 24, n°. 1, p. 132–147, 2012.

[76] W.-J. van Hoeve, “The alldifferent constraint : A survey,” *arXiv preprint cs/0105015*, 2001.

[77] K. Deb, “Multi-objective optimization,” dans *Search Methodologies : Introductory Tutorials in Optimization and Decision Support Techniques*, E. K. Burke et G. Kendall, édit. Boston, MA : Springer US, 2014, p. 403–449.

[78] Y. Dumas, J. Desrosiers, E. Gelinas et M. M. Solomon, “An optimal algorithm for the traveling salesman problem with time windows,” *Operations Research*, vol. 43, n°. 2, p. 367–371, apr 1995.

[79] G. Desaulniers, O. B. G. Madsen et S. Ropke, “The vehicle routing problem with time windows,” dans *Vehicle Routing : Problems, Methods and Applications*, 2e éd., P. Toth

et D. Vigo, édit. Philadelphia, PA : Society for Industrial and Applied Mathematics, 2014, ch. 5, p. 119–159.

[80] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Information Processing Letters*, vol. 1, p. 132–133, 1972.

[81] T. M. Chan, "Optimal output-sensitive convex hull algorithms in two and three dimensions," *Discrete & Computational Geometry*, vol. 16, n°. 4, p. 361–368, 1996.

[82] M. Shimrat, "Algorithm 112 : Position of point relative to polygon," *Communications of the ACM*, vol. 5, n°. 8, p. 434, 1962.

[83] H. Gehring et J. Homberger, "A parallel two-phase metaheuristic for routing problems with time-windows," *Asia Pacific Journal of Operational Research*, vol. 18, n°. 1, p. 35–48, 2001.

[84] SINTEF, "Gehring and Homberger benchmark," 2019, accessed : 2019-10-31. [En ligne]. Disponible : www.sintef.no/projectweb/top/vrptw/homberger-benchmark

[85] IBM ILOG CPLEX, "Optimization studio CP optimizer user's manual, version 12.8," 2017. [En ligne]. Disponible : www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/usrcpoptimizer.pdf