Matheurisc approaches for solving transport optimization problems in urban logistics

Dorian Dumez

HAL Id: tel-03394281

https://theses.hal.science/tel-03394281

Submitted on 22 Oct 2021

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS-DE-LA-LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies
de l'Information et de la Communication
Spécialité : *Informatique*

Par

Dorian DUMEZ

Approches matheuristiques pour la résolution de problèmes d'optimisation des transports en logistique urbaine

Matheuristic approaches for solving transport optimization problems in urban logistics

Thèse présentée et soutenue à Nantes, le 23 septembre 2021
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes (LS2N)
Thèse N° : 2021IMTA0253

Rapporteurs avant soutenance :

Claudia ARCHETTI   Professeur Associé à l'ESSEC Business School
Frédéric SEMET   Professeur à Centrale Lille

Composition du Jury :

Président :	Romain BILLOT	Professeur à IMT Atlantique
Examinateurs :	Claudia ARCHETTI	Professeure Associée à l'ESSEC Business School
	Romain BILLOT	Professeur à IMT Atlantique
	Frédéric SEMET	Professeur à Centrale Lille
	Samuel VERCRAENE	Maître de conférences à l'INSA Lyon
Dir. de thèse :	Fabien LEHUÉDÉ	Professeur à IMT Atlantique
Encadrant de thèse :	Olivier PÉTON	Professeur à IMT Atlantique

Invité(s) :

Nabil ABSI   Professeur à Mines Saint-Etienne
Stefan IRNICH Professeur à l'université Johannes Gutenberg de Mayence

RÉSUMÉ EN FRANÇAIS

Commençons par une définition traduite de Savelsbergh and Van Woensel (2016)[1] :

> ❝ *La logistique urbaine* consiste à trouver des moyens efficaces et efficients de transporter des marchandises dans les zones urbaines tout en tenant compte des effets négatifs sur la congestion, la sécurité et l'environnement. ❞

Cette thèse s'inscrit dans le vaste courant de travaux en recherche opérationnelle qui définissent des modèles et des méthodes pour améliorer le transport des marchandises sous de nombreux aspects. L'objectif de ce travail est de donner aux praticiens et aux universitaires un aperçu de politiques de distributions urbaines possibles, d'algorithmes d'optimisation pour ces dernières, et les simulations réalisées par ces algorithmes. Dans le cadre du projet OPUSS, nous avons travaillé en étroite collaboration avec la chaire de gestion logistique de la Gutenberg School of Management and Economics (Mayence, Allemagne), spécialisée dans les méthodes exactes pour les problèmes d'optimisation combinatoire. Nous nous concentrons sur les décisions opérationnelles de routage pour les petits commerces et les services privés dans les zones urbaines. En d'autres termes, nous cherchons à concevoir une planification quotidienne efficace pour les conducteurs, afin de remplir des tâches données à satisfaire dans une zone restreinte. A titre d'exemples, présentons brièvement quelques problèmes standards que nous allons étendre, recombiner et résoudre dans cette thèse :

— Le *problème de tournées de véhicules* (*Vehicle Routing Problem*, VRP, Dantzig and Ramser (1959)) cherche à trouver les itinéraires les moins coûteux afin de distribuer un ensemble donné de colis avec une flotte données de véhicules.

— Le *problème de tournées de véhicules avec fenêtre de temps* (*Vehicle Routing Problem with Time Windows*, VRPTW, Savelsbergh (1985)) est un extension du VRP où chaque client ne peut être servi que pendant une période prédéfinie, appelée *fenêtre de temps*.

— Le *problème de tournées de véhicules avec trajets multiples* (*Multi-Trip Vehicle Routing Problem*, MTVRP, Fleischmann (1990)) est une relaxation du VRP dans laquelle chaque véhicule est autorisé à effectuer plusieurs tournées successives dans la journée.

---

1. Traduction de l'anglais vers le français

— Le *problème de tournées de véhicules avec flux inverses* (*Vehicle Routing Problem with Backhauls*, VRPB, Gélinas et al. (1995)) est une extension du VRP où certaines marchandises doivent être livrées et d'autres marchandises collectées.

— Le *problème de tournées de véhicules généralisé* (*Generalized Vehicle Routing Problem*, GVRP, Ghiani and Improta (2000)) est une autre extension du VRP où les clients peuvent être servis à différents endroits.

— Le *problème de tournées de véhicules à deux échelons* (*2-Echelon Vehicles Routing Problem*, 2E-VRP, Crainic et al. (2009)) considère également la livraison de marchandises à moindre coût, mais via la coopération de deux flottes de véhicules. C'est-à-dire qu'une flotte de camions achemine d'abord les marchandises du dépôt, situé en périphérie, vers de petites plateformes logistiques proches du centre-ville, appelées *satellites*. Les colis sont ensuite livrés aux clients par de petits véhicules depuis les satellites.

Pour combiner de manière efficace et réaliste ces caractéristiques, il est important de coordonner les opérations des différents véhicules. Ainsi, des contraintes de synchronisation apparaissent dans les modèles. Celles que nous traitons dans cette thèse sont nommées *contraintes de ressource inter-tour* par Hempsch and Irnich (2008) ou *contraintes de synchronisation de ressources* par Drexl (2012) [2] qui les définies tels que :

> **❝** La consommation totale d'une ressource par tous les véhicules doit être inférieure ou égale à une limite spécifiée. **❞**

Au cours de cette thèse, nous définissons deux nouveaux problèmes d'optimisation. Tout d'abord, le problème de tournées de véhicules avec options de livraisons (*Vehicle Routing Problem with Delivery Options*, VRPDO, Dumez et al. (2021a); Tilk et al. (2020)) : il s'agit d'une extension du GVRPTW dans lequel chaque client propose quelques lieux de livraison possibles avec une fenêtre temporelle, correspondant aux lieux qu'il visitera dans la journée. Certains peuvent être des lieux de livraison partagés, comme des magasins ou des consignes, avec une capacité limitée partagée par tous les livreurs. Ensuite, le problème de tournées de véhicules à deux échelons avec trajets multiples, capacités aux satellites et flux inverses (*2-Echelon Multi-Trip Vehicle Routing Problem with Capacitated Satellites and Reverse Flow*, 2E-MTVRP-CSRF, Dumez et al. (2021b)) : il intègre le VRPTW et le VRPB dans le système logistique du 2E-VRP tout en considérant des satellites de capacité limitée. Ainsi, les véhicules des deux échelons doivent être synchronisés dans le temps et dans l'espace pour traiter les marchandises à livrer et à collecter, tout en veillant à ce que la quantité de marchandises en

---

2. Traduction de l'anglais vers le français

attente de transfert entre les deux échelons à un satellite ne dépasse jamais sa capacité.

Pour résoudre ces problèmes, nous utilisons des méthodes approchées, tandis que l'équipe de recherche de Mayence s'est concentrée sur les méthodes exactes. Glover and Kochenberger (2003)[3] définissent les *metaheuristiques* par :

> ❝ Des méthodes de résolution qui orchestrent des procédures d'amélioration locale et des stratégies de niveau supérieur pour créer un processus capable d'échapper aux optima locaux et d'effectuer une recherche robuste d'un espace de solution. ❞

De plus, dans le cadre d'une collaboration entre nos deux équipes, nous nous sommes intéressés de près aux *matheurisitiques*. Cette idée est décrite dans Fischetti and Fischetti (2018)[4] de la manière suivante:

> ❝ On peut insister sur l'approche de la programmation mathématique et essayer d'obtenir des résultats de plus en plus satisfaisants en améliorant le modèle et en enrichissant le solveur de fonctionnalités spécialisées. On peut aussi oublier la programmation mathématique et recourir à des heuristiques ad-hoc qui ne sont pas basées sur le modèle de programmation mathématique. Dans ce dernier cas, le modèle mathématique est complètement ignoré ou simplement utilisé pour illustrer les caractéristiques du problème [...] Une troisième approche est cependant possible, qui consiste à utiliser le solveur comme outil de base dans le cadre heuristique. Cette hybridation de la programmation mathématique avec des métaheuristiques conduit à l'approche matheuristique. ❞

Nous utilisons la métaheuristique de *recherche à voisinage large (Large Neighborhood Search, LNS)*. Son idée est d'améliorer itérativement la solution courante en la *ruinant* (c'est-à-dire en enlevant une partie) et en la *réparant* (c'est-à-dire en réinsérant les clients enlevées à un endroit adéquat). Ce processus est répété jusqu'à ce qu'un critère d'arrêt soit satisfait (généralement une limite de temps). Pour clarifier la position de notre travail, présentons un bref historique de LNS pour les problèmes de routage :

1. Shaw (1998) a initialement proposé LNS dans un contexte de programmation par contraintes pour résoudre le VRP. Dans cette première version, les solutions étaient réparées par un branch-and-bound utilisant un solveur de programmation par contraintes.

---

3. Traduction de l'anglais vers le français
4. Traduction de l'anglais vers le français

2. Schrimpf et al. (2000) a développé indépendamment une méthode proche sous le nom de *ruiner et recréer* (*ruin and recreate*) pour le VRPTW. Mais cette fois, les solutions étaient réparées par une méthode heuristique.

3. Pisinger and Ropke (2007); Ropke and Pisinger (2006a,b) ont popularisé LNS en proposant une méthode unifiée pour divers problèmes de routage (dont le VRPTW et le VRPB). En effet, Ropke and Pisinger (2006a) a maintenant été cité 1514 fois[5] ! Cette nouvelle version détruit et répare les solutions avec diverses heuristiques choisies de manière probabiliste en fonction de leurs performances passées. Ainsi, la méthode proposée est appelée recherche à voisinage large adaptative (Adaptive Large Neighborhood Search, ALNS).

4. Au cours des années suivantes, la méthode (A)LNS a été développée pour résoudre des problèmes variés et plus complexes (Pisinger and Ropke, 2019). Par exemple, pour le 2E-VRP, Hemmelmayr et al. (2012) ont développé plusieurs heuristiques travaillant sur différents aspects des solutions et Grangier et al. (2018) ont développé davantage les hybridations avec les solveurs de programmation mathématique.

5. A partir de ces nombreuses études, Santini et al. (2018) et Turkeš et al. (2020) ont réalisé des méta-analyses sur des composants spécifiques de l'ALNS, respectivement le critère d'acceptation et la couche adaptative. Ces études montrent que ce domaine de recherche commence à mûrir autour des connaissances structurées, comme l'espère Sörensen et al. (2018).

6. Christiaens and Vanden Berghe (2020) ont récemment proposé de réduire la méthode LNS à des composants plus simples. Ainsi, ils n'utilisent que quelques heuristiques de réparation simples et suppriment peu clients des solutions lors de chaque itération. Cela les conduit à une méthode capable d'effectuer un très grand nombre d'itérations très rapides.

Le premier axe de notre contribution méthodologique concerne la configuration de LNS. En effet, lors de notre revue de la littérature, de nombreuses heuristiques de destruction et de réparation nous ont semblé pertinentes. Nous avons conçu un protocole expérimental rigoureux basé sur des tests statistiques pour évaluer les redondances et les complémentarités entre les heuristiques. Ceci nous a permis de décider d'une configuration restreinte et efficace de LNS. Par ailleurs, au cours de ces nombreuses expériences, nous avons remarqué que, comme suggéré par Christiaens and Vanden Berghe (2020), des destructions limitées et une reconstruction rapide conduisent à de bons résultats. Par conséquent, nous proposons une configuration LNS

---

5. Selon Google scholar le 13/04/2021

qui combine des destructions de petite et de grande taille. L'algorithme *SLNS* (Small and Large Neighborhood Search) ainsi développé s'est avéré être un algorithme performant sur quatre problèmes généralisés de routage de véhicules avec fenêtres de temps.

Le deuxième axe de notre contribution méthodologique concerne l'orchestration de LNS avec des composants exacts, créant ainsi des matheuristiques. Des couches adaptatives ont été développées pour décider du moment où il faut utiliser ces outils. En effet, ces outils peuvent améliorer significativement les résultats de LNS. En revanche, ils peuvent être trop lents à des moments où LNS aurait obtenu de meilleurs résultats, tel que des phases d'intensification intensive rapide. Au cours de cette thèse, nous avons développé des méthodes entrant dans chaque classe de la classification des matheuristiques de Archetti and Speranza (2014) :

Approches basées sur la génération de colonnes : un modèle basé sur les routes a été écrit pour le VRPDO et est résolu régulièrement par un solveur afin de recombiner les routes produites par LNS au cours de ses itérations.

Improvement heuristics : une approche de programmation dynamique a été adoptée pour améliorer les routes prometteuses produites par LNS pour le VRPDO. Nous avons adapté le voisinage de Balas-Simonetti (Balas, 1999).

Approches de décomposition : la méthode développée pour résoudre le 2E-MTVRP-CSRF est basée sur une décomposition en trois sous-problèmes. Cette méthode alterne entre le travail sur le premier échelon avec SLNS, le travail sur le second échelon avec SLNS, et la résolution d'un modèle pour recombiner les deux échelons.

La présentation des deux nouveaux problèmes et des deux contributions méthodologiques est organisée comme suit :

Chapitre 1 il introduit le problème de tournées de véhicules avec options de livraison (VRPDO, Dumez et al. (2021a); Tilk et al. (2020)) ainsi que la recherche par petits et grands voisinages (SLNS) développée pour le résoudre. Nous questionnons rigoureusement l'impact de chaque composant.

Chapitre 2 il présente l'hybridation du SLNS avec des solveurs MIP et la programmation dynamique pour résoudre une variété de problèmes généralisés de tournées de véhicules avec fenêtres temporelles. L'utilité de chaque outil et les règles pour les orchestrer sont longuement étudiées dans le contexte de quatre variantes de problème de tournées de véhicules généralisé.

Chapitre 3 il présente le problème de tournées de véhicules à deux échelons avec trajets multiples, capacités sur les satellites et flux inverses (2E-MTVRP-CSRF, Dumez et al. (2021b))

et la méthode de décomposition développée pour le résoudre. Cette méthode alterne entre le travail sur chaque échelon avec SLNS et la résolution d'un modèle mathématique par un solveur pour recombiner les deux échelons. Ainsi, ce chapitre réutilise les développements précédents pour résoudre un problème complexe.

Les deux premiers chapitres sont basés sur les articles Dumez et al. (2021a) et Dumez et al. (2021c) publiés dans Transportation Research part B et EURO Journal in Transportation and Logistics. Ces résultats ont été présentés dans les conférences suivantes : Dumez et al. (2019b), Dumez et al. (2019a), Péton et al. (2019), Dumez et al. (2019c), Dumez et al. (2020). Le dernier chapitre est basé sur l'article Dumez et al. (2021b), qui sera soumis prochainement à une revue internationale.

ACKNOWLEDGEMENT

" End? No, the journey doesn't end here. "

Je tiens à remercier Professeur Fabien Lehuédé et Professeur Olivier Péton pour la supervision de ma thèse.

J'aimerais remercier Professeur Stefan Irnich, Christian Tilk, et Katharina Olkis pour leurs collaborations durant ma thèse.

Je voudrais remercier mes collègues à l'IMT Atlantique avec qui j'ai passé ces 3 ans, ainsi que toute la chair en management logistique de l'université de Mayence qui m'a accueilli pendant un mois.

Je remercie aussi le jury pour leur lecture attentive de ce manuscrit.

Enfin, et bien évidemment, je remercie toute ma famille et mes amis.

---

[1] From "The Lord of the Rings: The Return of the King" (Jackson et al., 2003)

TABLE OF CONTENTS

INTRODUCTION

Let us start with a definition from Savelsbergh and Van Woensel (2016):

> ❝ *City Logistic* is about finding efficient and effective ways to transport goods in urban areas while taking into account the negative effects on congestion, safety, and environment. ❞

This thesis is part of the broad stream of studies in operations research which are defining models and methods to improve the transportation of goods over many aspects. The goal of this work is to give practitioners and academics an overview of possible urban distribution policies, optimisation algorithms for these, and simulations performed by these algorithms. In the frame of the OPUSS project, we worked in tight collaboration with the chair of logistics management of the Gutenberg School of Management and Economics (Mainz, Germany) which is specialized in exact methods for combinatorial optimization problems. We focus on the operational routing decisions for small shops and private services in urban areas. That is to say, we seek to find efficient day-to-day routes and planning for drivers in order to fulfill given tasks to perform in a restricted zone. As examples, let us briefly present a few standard problems that we extend, recombine and solve in this thesis:

— The *Vehicle Routing Problem* (VRP, Dantzig and Ramser (1959)) seeks to find the least-cost routes in order to distribute a given set of parcels with a given fleet of vehicles

— The *Vehicle Routing Problem with Time Windows* (VRPTW, Savelsbergh (1985)) is a extention of the VRP where each customer can be served solely during a pre-defined period, called *time window*.

— The *Multi-Trip Vehicle Routing Problem* (MTVRP, Fleischmann (1990)) is a relaxation of the VRP in which each vehicle is allowed to performed multiple successive tours during the day.

— The *Vehicle Routing Problem with Backhauls* (VRPB, Gélinas et al. (1995)) is an extension of the VRP where some goods have to be delivered in addition to some goods to be collected.

— The *Generalized Vehicle Routing Problem* (GVRP, Ghiani and Improta (2000)) is another extension of the VRP where the customers can be served at different places, to be chosen

by the delivery company.

— The *2-Echelon Vehicle Routing Problem* (2E-VRP, Crainic et al. (2009)) also considers the least-cost delivery of goods, but via the cooperation between two fleets of vehicles. That is to say, a fleet of trucks first bring goods from the depot located in the outskirts to small logistic platforms near the city center, called *satellites*. The parcels are then delivered to the customers by small vehicles from the satellites.

In order to combine these features in an efficient and realistic way, it is important to coordinate the operations of the different vehicles. Therefore, synchronization constraints arise in the models. Those we treat in this thesis are named *inter-tour resource constraint* by Hempsch and Irnich (2008) and as *resources synchronization constraint* by Drexl (2012), which propose the following definition :

> ❝ The total consumption of a specified resource by all vehicles must be less than or equal to a specified limit. ❞

We define two new problems during this thesis. First, the Vehicle Routing Problem with Delivery Options (VRPDO, Dumez et al. (2021a); Tilk et al. (2020)): it is a refinement of the GVRPTW in which each customer gives several possible locations for delivering its order along with a time windows, corresponding to places he/she will visit during the day. Some can be shared delivery locations, such as shops or lockers, with a limited capacity to be used by all deliverymen. Second, the 2-Echelon Multi-Trip Vehicle Routing Problem with Capacitated Satellites and Reverse Flow (2E-MTVRP-CSRF, Dumez et al. (2021b)): it integrates the VRPTW and the VRPB in the 2E-VRP logistic system while considering satellites of limited capacity. Thus, the vehicles of the two echelons must be synchronised in time and space to handle the goods to be delivered and collected, while integrating that the quantity of goods waiting to be transferred between the two echelons at a satellite never exceeds its capacity.

To solve these problems we use approximate methods while the Mainz's research team focus on exact methods. Glover and Kochenberger (2003) defines a *metaheuristic* such as:

> ❝ Solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space. ❞

Moreover, in the context of a collaboration between our two teams, we take a great interest in *matheurisitics*. This idea is described in Fischetti and Fischetti (2018) in the following manner:

❝ One can insist on the mathematical programming (MP) approach and try to obtain better and better results by improving the model and by enhancing the solver by specialized features. Or one can forget about MP and resort to ad-hoc heuristics not based on the MP model. In this latter case, the MP model is completely disregarded or just used for illustrating the problem characteristics [...] A third approach is however possible that consists in using the MP solver as a basic tool within the heuristic framework. This hybridization of MP with metaheuristics leads to the matheuristic approach. **❞**

We use the *Large Neighborhood Search (LNS)* metaheuristic. Its idea is to iteratively improve the current solution by ruining it (i.e. removing a part of it) and repairing it (i.e reinserting the removed customers at appropriated places). This process is repeated until a stopping criterion (usually a time limit). To clarify the position of this work, let us present a brief history of LNS for routing problems:

1. Shaw (1998) initially proposed LNS in a constraint programming context to solve the VRP. In this first version, the solutions were repaired by a branch-and-bound using a constraint programming solver.

2. Schrimpf et al. (2000) independently developed a very similar method under the name *ruin and recreate* for the VRPTW. But this time the solutions were repaired by one heuristic method.

3. Pisinger and Ropke (2007); Ropke and Pisinger (2006a,b) popularized LNS by proposing an unified method for various routing problems (including the VRPTW and the VRPB). Indeed Ropke and Pisinger (2006a) is now cited 1514 times [7] ! This new version destroys and repairs the solutions with various heuristics chosen probabilistically based on their past performance. This method is called Adaptive Large Neighborhood Search (ALNS).

4. During the following years, (A)LNS was further developed to solve various and more complex problems (Pisinger and Ropke, 2019). For example, for 2E-VRPs, Hemmelmayr et al. (2012) developed various heuristics working on different aspects of the solutions and Grangier et al. (2018) integrated time windows and further developed the hybridizations with MIP solvers.

5. From these numerous studies, Santini et al. (2018) and Turkeš et al. (2020) conducted meta-analysis on specific components of ALNS, respectively the acceptance criterion and the adaptive layer. These studies show that this field of research is starting to mature around structured knowledge, as hoped by Sörensen et al. (2018).

---

7. According to Google scholar the 13/04/2021

6. Christiaens and Vanden Berghe (2020) recently proposed to narrow down the LNS method to simpler components. Thus, they use only a few simple repair heuristics and remove only a few customers from the solutions during each iteration. This leads them to a method capable of performing a tremendous number of very fast iterations. The suggested philosophy for LNS proved to be very competitive on the VRP and the VRPTW.

The first axis of our methodological contribution is about the configuration of LNS. Indeed, during our literature review many ruin and recreate heuristics seemed relevant. We designed a rigorous experimental protocol based on statistical tests to evaluate redundancies and complementarities between heuristics. This allowed us to decide on a restricted and efficient configuration of LNS. Besides, during these many experiments, we noticed that, as suggested by Christiaens and Vanden Berghe (2020), limited destructions and fast reconstruction lead to good results. As a result, we propose a LNS configuration that combines small and large destruction sizes. The resulting developed *Small and Large Neighborhood Search (SLNS)* proved to be a state of art algorithm on four generalized vehicle routing problems with time windows.

The second axis of our methodological contribution is about the orchestration of LNS with exact components, thus creating matheuristics. Adaptive layers were developed to decide on when to use these tools. Indeed, these tools can significantly improve the results on LNS. On the other hand, they may be too slow at times when LNS would have performed better on its own, ushc as during fast intensification phases. In the course of this thesis we developed methods falling into each class of Archetti and Speranza (2014) matheuristics classification:

Column generation-based approaches a route-based model is written for the VRPDO to be solved regularly by a MIP solver in order to recombines routes produced by LNS in the course of its iterations.

Improvement heuristics a dynamic programming approach is taken to improve promising routes produced by LNS for the VRPDO. Namely, we adapt the Balas-Simonetti neighborhood (Balas, 1999).

Decomposition approaches the method developed to solve the 2E-MTVRP-CSRF is based on a threefold decomposition of the problem. This method alternates between working on the second echelon with SLNS, working on the first echelon with SLNS, and solving a MIP to recombine the two echelons.

The presentation of the two new problems and of the two methodological contributions is organized as follow:

Chapter 1 introduces the Vehicle Routing Problem with Delivery Options (VRPDO, Dumez

et al. (2021a); Tilk et al. (2020)) together with the Small and Large Neighborhood Search (SLNS) developed to solve it. We rigorously question the impact of each component of the developed method and the impact of the proposed parameters.

Chapter 2 presents the hybridization of the aforementioned SLNS with a column-generation-based approach and dynamic programming to solve a variety of Generalized Vehicle Routing Problems with Time Windows. The usefulness of each tool and the rules to orchestrate them is extensively studied in the context of four variants of the GVRPTW.

Chapter 3 presents the 2-Echelon Multi-Trip Vehicle Routing Problem with Capacitated Satellites and Reverse Flow (2E-MTVRP-CSRF, Dumez et al. (2021b)) and the decomposition method developed to solve it. This method alternates between working on the each echelon with SLNS and solving a MIP with a commercial solver to recombine the two echelons. Thus, this chapter reuses the previous developments to solve a complex problem.

The first two chapters are a streamlines and updated versions of the articles Dumez et al. (2021a) and Dumez et al. (2021c) published in Transportation Research part B and Euro Journal in Transportation and Logistics. These results were presented in the following conferences: Dumez et al. (2019b), Dumez et al. (2019a), Péton et al. (2019), Dumez et al. (2019c), Dumez et al. (2020). The last chapter is based on the article Dumez et al. (2021b), which is to be submitted soon to an international journal.

LARGE NEIGHBORHOOD SEARCH HEURISTICS FOR THE VEHICLE ROUTING PROBLEM WITH DELIVERY OPTIONS

1.1 Introduction

The revenue of online stores in France was 92.6 billion euros in 2018, after an average growth of 13% per year during the last four years (Moyou, 2019). With this growth of e-commerce, an increasing number of parcels have to be delivered each day. Consequently, several possibilities have been developed so that distribution can be faster and cheaper. According to Morganti et al. (2014), in 2012, 90% of the French and German population were within 10 minutes of a locker or a pick-up point (often shops or post offices), and the number of such locations grew by 33% between 2008 and 2012. Furthermore, trunk deliveries were recently tested on an industrial scale (McFarland, 2018).

The customer is generally required to choose one delivery location. Nevertheless, people move around during the day, for example, to go to work or take children to school. This is the major cause of no-show in attended deliveries. According to Allen et al. (2016), in the UK, 14% of all deliveries fail. In 2014, in the UK, the cost of these failed deliveries has been estimated at £771 million. Consequently, it is very likely that a purchaser would like to choose between several delivery options depending on the time of delivery.

The objective of carriers is to deliver all their parcels at a minimum cost. They also want to maintain a good quality of service, but the time window width has a great impact on delivery costs. Nevertheless, they are essential because 80% of the parcels delivered in the UK do not fit into the letterbox (Allen et al., 2016) and nobody likes to wait all afternoon at home for a parcel (Agatz et al., 2008). Accordingly, new strategies are developed to increase the number of successful deliveries (Florio et al., 2018).

This chapter discusses the *Vehicle Routing Problem with Delivery Options (VRPDO)*. It is an extension of both the Vehicle Routing Problem with Time Windows (VRPTW, Savelsbergh (1985)) and the Generalized Vehicle Routing Problem (GVRP, Ghiani and Improta (2000)). In the VRPDO, each customer can choose to have a package delivered through several delivery options. For example, a given parcel can be delivered at office during work hours, at home in the evening or in a locker at any time.

From the standpoint of the carrier, these delivery options can be of two natures. They can take place in individual delivery locations, like a home or a car trunk. Only one customer can be delivered in that location. Otherwise, deliveries can take place at *shared delivery locations (SDL)*, like lockers or pick-up points. In this case, several parcels can be left at the same location, while possibly satisfying some capacity constraints.

The contribution of this chapter is two-fold. First, from the managerial point of view, a

new variant of the Vehicle Routing Problem (VRP) is introduced: the VRPDO combines delivery options to reduce delivery costs while guaranteeing a higher quality of service. From the methodological point of view, a Large Neighborhood Search (LNS) metaheuristics that embeds many recent and new ideas is proposed. This LNS includes many operators from the literature. We compare them rigorously to determine an efficient and non-redundant configuration. We introduce a new LNS framework which combines small and large destructions: the *Small and Large Neighborhood Search (SLNS)*. We show that this framework is particularly efficient to escape local optima in case of complex constraints such as the synchronization constraints of the VRPDO.

This chapter is structured as follows: In Section 1.2, the VRPDO is described and a mathematical model is proposed. Section 1.3 exposes the literature on related problems. Section 1.4 details the first LNS developed to solve the VRPDO. In addition, our methodology to configure the algorithm is explained. Section 1.5 present the SLNS. Finally, sections 1.6 and 1.7 develops experiments, first to validate the method and second to draw managerial insights from randomly generated instances of the VRPDO.

1.2 The Vehicle Routing Problem with Delivery Options

This section details all components of the VRPDO (Section 1.2.1) and gives a mathematical model (Section 1.2.2).

This problem was defined in collaboration with the Mainz research team. It is described in Dumez et al. (2021a) and Tilk et al. (2020). The former develops a heuristic solution method while the latter describes an exact algorithm.

1.2.1 Problem settings

The VRPDO is an operational optimization problem defined on a short time horizon, typically one day. In this problem, each customer can choose several *delivery options*. In the following, for the sake of conciseness, we will refer to them only as *options*.

An option is a tuple associated with a unique customer, composed of *a location*, *a preference level* and *a service duration,* as follows:

— A *location* has a geographical address and a time window during which goods can be delivered. For each location, a fixed preparation time represents the time needed to park a vehicle and access the delivery point. Two types of locations are considered: individual delivery locations and shared delivery locations (SDLs). An individual delivery location is typically a personal address. Only one option can be associated with this location. It generally has a tight time window. An SDL is typically a pickup point or a locker. Several options can be associated with this location. An SDL can be given a capacity, which is defined as the maximum number of packages which can be delivered at this location during the time horizon. For the sake of conciseness, SDLs and individual delivery locations may be called "shared locations" and "individual locations", respectively.

— The *preference level* of an option is an integer in $\{1, ..., \bar{p}\}$, $\bar{p}$ being the number of preference levels. 1 is the value associated with the customer's preferred option and $\bar{p}$ corresponds to the least preferred option.

— The *service duration* of an option represents the time needed to deliver the customer's package after accessing the location.

Let us consider a homogeneous fleet of vehicles starting from a given depot and returning to this depot within the time period. Travel time and routing costs between each pair of locations are assumed to be known.

The VRPDO consists in designing a route for each vehicle, serving each customer with one of his/her options, and satisfying the SDL capacities and minimal service level constraints. For each level $p \in \{1, ..., \bar{p} - 1\}$, the service level SL_p of a solution is defined as the percentage of customers served with an option of level at most p. A minimal service level β_p has to be achieved for each level $p \in \{1, ..., \bar{p} - 1\}$.

Similarly to many vehicle routing problems, the objective function is the minimization of the number of vehicles and the sum of routing costs, in lexicographical order. Note that each customer's order has to be delivered by a single truck (i.e. split deliveries are not allowed).

When a customer orders a parcel, he/she chooses multiple options, sorted by preference level. In the VRPDO, the time windows relate to the delivery of parcels by vehicles. At individual locations, these time windows are selected by customers. At SDLs, they correspond to an interval of time during which the corresponding facilities can be accessed by vehicles. Thus, it is assumed that all options delivered at SDLs have the same time windows. In practice, once a customer's order is delivered at an SDL or a remote location (i.e. car's trunk), the customer is notified ; he/she can collect the package at any time after being notified, including on the next days. Hence, once an order is delivered at an SDL, it is considered to occupy a part of the SDL capacity until the end of the period. There might be several policies to manage uncollected parcels, but this question is out of the scope of the present study. Thus, we simply assume that the capacity of an SDL is *a priori* adjusted before solving the VRPDO, according to the parcels remaining at this location.

To illustrate the definition of the VRPDO, Figure 1.1 shows the options from the customer perspective. Figure 1.1 shows the example of a customer with 3 delivery options. Delivery at home between 7 p.m. and 9 p.m. is the preferred option, then delivery in a locker between 1 p.m. and 5 p.m., then a delivery in the trunk of his/her car between 9 a.m. and 6 p.m.

Figure 1.2 presents the options from the carrier perspective. The individual options of each customer are represented by a group of icons of identical colors. SDLs are represented by black icons (locker 🔒, store/postal office 🏤. Each customer is represented by a number and its options are surrounded by a line of the same color as the icons. The carrier has to serve all customers via exactly one option. This amounts to define a set of routes that start from the carrier's depot (represented by the truck icon 🚚) and visit exactly one option inside a colored line.

Like in the Generalized Vehicle Routing Problem (GVRP, Ghiani and Improta (2000)), the set of options is composed of subsets, for each customer. Vehicles must visit exactly one option in each subset. In the GVRP, subsets of options form a partition of the set of locations. In the VRPDO, the subsets of options form a cover of the set of locations. This is because options

Address	Time	Preference
My home	[19:00;21:00]	1
My office	[9:00;18:00]	2
My locker	[0:00;24:00]	3
My pickup point	[0:00;24:00]	3

Figure 1.1 – Example of a customer's preferences in the VRPDO
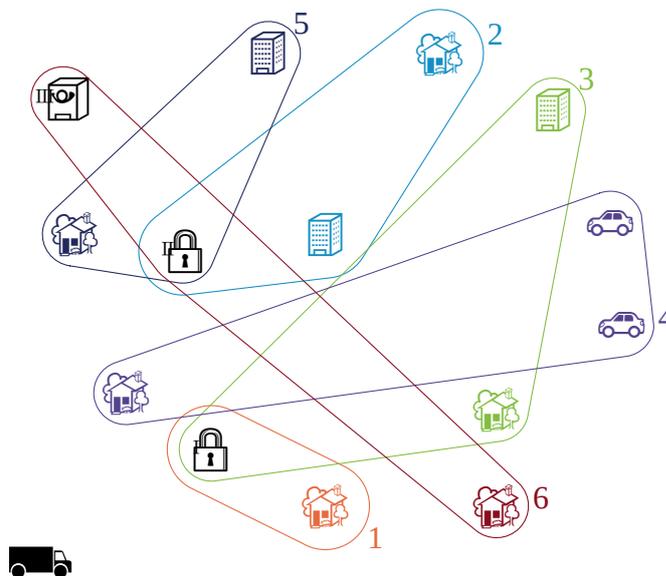
Figure 1.2 – An instance of the VRPDO from the carrier perspective

associated with distinct customers can take place at the same shared location. An instance of the Generalized Vehicle Routing Problem with Time Windows (GVRPTW, Moccia et al. (2012)) is an instance of the VRPDO without shared locations and a single preference level. Hence, the VRPDO can be seen as a generalization of the GVRPTW.

Figure 1.3 represents a solution to the VRPDO instance introduced in Figure 1.2, with only one route. An option that takes place in an individual location is depicted by a circle including the customer's identity and the location. This is the case of customer 2, who is delivered at office rather than at home, thus avoiding a big detour. A shared location is depicted by a rectangle that

includes the locations and identifies the delivered customers. For example, customers 1 and 3 are delivered at SDL I.
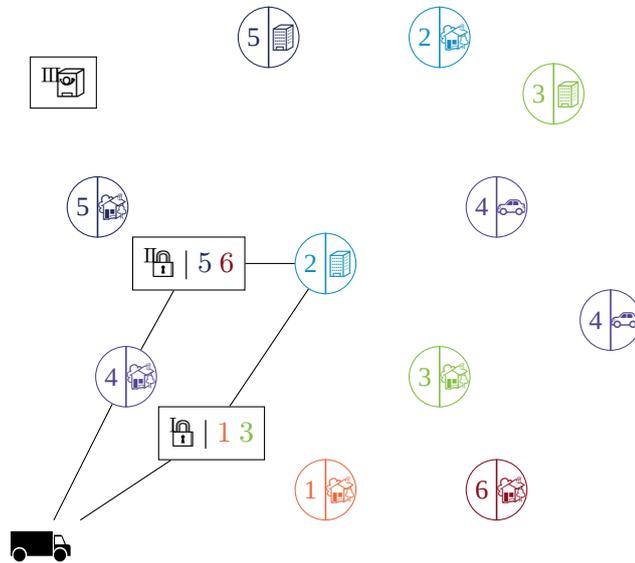
Figure 1.3 – Example of a route in the VRPDO

The time spent by a vehicle at a location is equal to the preparation time of the location plus the service duration of the visited options. The preparation time of a location corresponds to the time to find a parking place. Hence, it is counted only once, regardless of the number of parcels delivered. The service duration of an option corresponds to the time to deliver the parcel or to put it in a box.

Figure 1.4 represents a solution to the VRPTW for the same instance, using home delivery only.

We assume that a parcel always occupies one box in a locker, no matter its size. Indeed, a locker is a rack of automatic boxes so there is at most one parcel per box, as long as it fits. Hence, the capacity of the shared locations, generally expressed as a number of boxes, can equivalently be expressed as a number of parcels. Due to uncertainty on when customers will pick up their parcels, it is assumed that each box can be used only once per day. On the contrary, it is realistic to consider that some shared locations (e.g. post offices) have unbinding capacity.

The assignment of parcels to boxes is not taken into account in the VRPDO, but it would be possible to consider different sizes of boxes by duplicating options.

Figure 1.5 illustrates the capacity constraints induced by shared locations for the example described in Figure 1.3. Not all shared locations must be visited. On the contrary, they can be visited by different vehicles.

Figure 1.4 – Example of a route in the VRPTW, with home delivery only

Figure 1.5 – Capacity at shared delivery locations (SDL)

Considering the quality of the service, Figure 1.6 represents an example with a service level constraint requiring that at least $\beta_1 = 50\%$ of the customers are served with their preferred options and that at least $\beta_2 = 75\%$ of the customers are served with options of level 1 or 2. In the example, 50% of the customers are served with their preferred option (SL_1) and 83% of the customers are served with an option of level 1 or 2 (SL_2), so that the service level constraints are satisfied.

The constraints related to SDL capacity and service level constraints can be seen as resource constraints involving all the vehicles. They fall under the category of *synchronized resources* defined by Drexl (2012) as: "At any point in time, the total utilization or consumption of a specified resource by all vehicles must be less than or equal to a specified limit".

Figure 1.6 – Service level measured with respect to a set of selected options

1.2.2 Mathematical model

Let N be the set of customers and O the set of options. The options of customer $c \in N$, denoted by $O_c \subset O$, are specific to that customer only. Thus, $\bigcup_{c \in N} O_c = O$ and $\forall c, c' \in N : c \neq c' \Rightarrow O_c \cap O_{c'} = \emptyset$.

Let L denote the set of locations and $O_l \subset O$ be the subset of options that take place at location $l \in L$.

The VRPDO is modeled on an option-based complete graph $G = (V, A)$. The set of vertices, denoted by $V = O \cup \{0, 0'\}$, contains one vertex for each option plus the starting depot 0 and the ending depot $0'$. The routes are defined on the option-based graph G. A route is a sequence of options belonging to distinct customers. Consequently, when multiple deliveries are made in a shared location, it is represented by several vertices of V. Figure 1.7 is the representation of the route from of Figure 1.3.

Figure 1.7 – Representation of the route from Figure 1.3 on Graph G

The travel time on arc $(i, j) \in A$ is denoted by $t_{i,j}$. It includes the preparation time at the location of option j if the two locations are distinct. The cost of traveling on arc (i, j) is denoted by $c_{i,j}$. The service duration s_i at vertex $i \in V$ represents the time necessary to visit the option associated with this vertex. It is assumed to be null at depot vertices 0 and $0'$. Each option $i \in O$ is associated with the time window $[a_i, b_i]$ of its location. Moreover, it is assumed that $a_0 = a_{0'} = 0$ and that b_0 and $b_{0'}$ correspond to the end of the working day.

Let $\bar{p}$ be the number of preference levels. For a preference level $p \in \{1, ..., \bar{p}\}$, β_p is the minimal percentage of customers that can be served with an option of level p or lower. By definition $\beta_P = 100\%$ and $\forall p \in \{1, ..., \bar{p} - 1\} : \beta_p \leqslant \beta_{p+1}$.

Let C_l be the capacity of location $l \in L$, expressed as a number of parcels. Each option $o \in O$ is associated with a preference level p_o and a demand q_o which is equal to the demand of its associated customer. The set of vehicle is denoted by K. It is assumed that the fleet is homogeneous and the capacity of vehicles is denoted by Q.

Model 1 describes the VRPDO. $x_{i,j}^k$ is a binary variable that indicates if the vehicle $k \in K$ uses arc $(i,j) \in A$. y_o is a binary variable that states whether option $o \in O$ is visited. h_i^k is the service time of vehicle $k \in K$ at vertex $i \in V$.

The first objective function (1.a) minimizes the number of vehicles. The second objective function (1.b) minimizes the routing costs. Constraints (1.c) state that exactly one option must be selected for each customer. Constraints (1.d) represent the satisfaction of vehicle capacity. Recalling that the capacity of SDLs is expressed as a number of parcels, independently of their size, constraints (1.e) state that the number of options delivered at location $l \in L$ cannot exceed its capacity C_l. Constraints (1.f) model the satisfaction of preference levels. For each preference level $p \in \{1, ..., \bar{p}\}$, at least $\beta_p \%$ of the total number of customers must be served with an option of level p or better. Constraints (1.g) state that an option can be used only if it is visited by one vehicle. The remaining constraints are the classical VRPTW constraints (Desaulniers et al., 2014): Constraints (1.h) ensure that a vehicle that leaves the depot returns it at the end of its route, Constraints (1.i) ensure the continuity of the routes, Constraints (1.j) compute the service time and Constraints (1.k) ensure the respect of the time windows. M is an arbitrarily large positive value.

Model 1: option-based model

$$lex - min(z_1, z_2)$$

$$s.t.\ z_1 = \sum_{j \in V} x_{0,j}^k \tag{1.a}$$

$$z_2 = \sum_{k \in \{1,...,K\}} \sum_{(i,j) \in A} c_{i,j} x_{i,j}^k \tag{1.b}$$

$$\sum_{o \in O_c} y_o = 1 \qquad \forall c \in N \tag{1.c}$$

$$\sum_{o \in O} \sum_{(i,o) \in A} x_{i,o}^k q_o \leqslant Q \qquad \forall k \in K \tag{1.d}$$

$$\sum_{o \in O_l} y_o \leqslant C_l \qquad \forall l \in L \tag{1.e}$$

$$\sum_{o \in O | p_o \leqslant p} y_o \geqslant \beta_p \times |N| \qquad \forall p \in \{1, ..., \bar{p} - 1\} \tag{1.f}$$

$$\sum_{(i,o)\in A}\sum_{k\in K} x_{i,o}^k \geqslant y_o \qquad\qquad \forall o \in O \qquad \text{(1.g)}$$

$$\sum_{j\in V} x_{0,j}^k = \sum_{i\in V} x_{i,0'}^k \qquad\qquad \forall k \in K \qquad \text{(1.h)}$$

$$\sum_{(i,j)\in A} x_{i,j}^k - \sum_{(j,i)\in A} x_{j,i}^k = 0 \qquad\qquad \forall k \in K, \forall j \in O \qquad \text{(1.i)}$$

$$h_i^k - h_j^k + t_{i,j} + s_i \leqslant M.(1 - x_{i,j}^k) \qquad \forall k \in K, \forall (i,j) \in A \qquad \text{(1.j)}$$

$$a_i \leqslant h_i^k \leqslant b_i \qquad\qquad \forall k \in K, \forall i \in V \qquad \text{(1.k)}$$

$$x_{i,j}^k \in \{0,1\} \qquad\qquad \forall k \in K, \forall (i,j) \in A$$

$$y_o \in \{0,1\} \qquad\qquad \forall o \in O$$

$$h_i^k \geqslant 0 \qquad\qquad \forall k \in K, \forall i \in V$$

1.3 State of the art

The VRPDO is a VRPTW with multiple delivery options and synchronized resources. This section is organized according to these three aspects: VRPTW, delivery options and resource synchronization. In addition, some closely related problems are laid out.

1.3.1 VRPTW

Since the introduction of the VRPTW by Savelsbergh (1985), a plethora of papers have been published on this subject. We refer to Bräysy and Gendreau (2005a,b) for a review of the applicable literature and to Vidal et al. (2013) for the latest advances. Many LNS heuristics have been used to solve problems related to the VRPTW. They will be briefly discussed in Section 1.4.

1.3.2 Delivery options

In the VRPDO, customer requests can be satisfied at various locations. This characteristic is shared by the *Generalized Vehicle Routing Problem with Time Windows* (GVRPTW, Moccia et al. (2012)). The GVRPTW is a specific case of the VRPDO without shared locations and with a single preference level. The literature on the GVRPTW is very scarce. We refer to Bektaş et al. (2011) and Afsar et al. (2014) for details on the GVRP, and to Moccia et al. (2012) for details of the GVRPTW. Moccia et al. (2012) proposed a tabu search able to tackle instances with 120 clusters within few minutes. Yuan et al. (2020a) and Yuan et al. (2020b) developed models and a branch-and-cut algorithm for the GTSPTW (a single vehicle GVRPTW) that can solve instances with 30 clusters. This method is extended in Yuan (2019) and Yuan et al. (2021) for the GVRPTW as a heuristic-based branch-and-cut. This method provides high quality solutions on instances with up to 120 customers in a few hundred seconds.

The *Vehicle Routing Problem with Multiple Time Windows* (VRPMTW, Favaretto et al. (2007)) can be seen as a special case of the VRPDO and of the GVRPTW, where all options of a customer take place at the same location and their time windows are disjointed. For more detail, see Tricoire et al. (2010), Belhaiza et al. (2014) and Hoogeboom et al. (2020).

In addition to the GVRP, some classes of vehicle routing problems include choices in the locations to visit. The goal of these problems is to select locations to visit and find routes between these locations, such that requirements are met at minimum cost or such that profit is

maximized. This category of problem includes the *Team Orienteering Problem (TOP)*, the *VRP with profit* (Archetti et al., 2014; Vansteenwegen et al., 2011), the *Traveling Purchaser Problem* (TPP, Bernardino and Paias (2018); Gendreau et al. (2016)), and the *Covering Tour Problem* (CTP, Kammoun et al. (2017)). For the interested reader, Yuan (2019) broadly discuss *non-Hamiltonian routing problems*.

Among the heuristics that have been proposed to solve VRP with choices in locations or time windows, most of the authors include a perturbation component in their method. For variants of the TOP, Hu and Lim (2014) developed a simulated annealing heuristic using ruin-and-recreate perturbations and a set partitioning formulation, Souffriau et al. (2013) combined the Greedy Randomized Adaptive Search Procedure (GRASP) with an Iterated Local Search (ILS) and Tricoire et al. (2010) used a Variable neighborhood Search (VNS) metaheuristic. For variants of the CTP, Allahyari et al. (2015) combined GRASP with ILS, Takada et al. (2015) used ILS, Vargas et al. (2015) worked with an Adaptive Large neighborhood Search (ALNS) and Kammoun et al. (2017) proposed a VNS.

Besides, some papers propose using dynamic programming. In the context of a multi-period TOP with multiple time windows, Tricoire et al. (2010) used dynamic programming to choose which time window should be used in a given route. Moccia et al. (2012) solved a GVRPTW; when a customer is inserted in a route, the options for the other customers of this route can be changed through dynamic programming. Vargas et al. (2015) solved the CTP with an ALNS algorithm that uses dynamic programming to decompose a giant tour into routes.

1.3.3   Resource synchronization

Drexl (2012) underlined that Hempsch and Irnich (2008) was almost the only paper considering synchronized resources. Few other heuristics have been proposed since then. In Hempsch and Irnich (2008), Grangier et al. (2017), Froger et al. (2017), the resources are only temporarily used during a limited period of time, before becoming available again. For example, in Grangier et al. (2017), a truck uses a dock at a satellite facility only during the loading and unloading operations.

In the VRPDO, the synchronized resources are permanently used during the whole time horizon. Thus, it is only necessary to count the number of parcels in each locker and the number of visited options of each preference level to check the satisfaction of the synchronized resource constraints.

Souffriau et al. (2013) defined a variant of the TOP applied to tour planning for tourists. It is

called the Multiconstraint Team Orienteering Problem with Multiple Time Windows (MC-TOP-MTW). The routes represent the different days of the trip. The synchronized resource constraints represent the budget and the maximal number of monuments of each type that can be visited. These resources are considered in a local search with a label on the locations. These labels are updated at each modification of the schedule. To get good solutions even with these resources, their algorithm relies on perturbations through a combination of GRASP with ILS. Besides, local search moves are chosen with a score. It is a function of the increase in the objective function, time shift, and resource consumption.

1.3.4 Closely related problems

Now that the three components of the VRPDO have been described, let us focus on very close problems.

Reyes et al. (2017) defined the *VRP with Roaming Delivery Locations (VRPRDL)*. This problem involves delivering parcels into the trunks of cars, which move according to known schedules. In the VRPRDL, there are no synchronized resources, the time windows of a customer are disjointed and cars do not move faster than the delivery truck. This problem is solved with an LNS heuristic. Classical operators are adapted and dynamic programming is used to re-optimize routes by selecting better options without changing the customer sequence in the route. The combination with home delivery is considered in a variant called *VRP with Home and Roaming Delivery Locations (VRPHRDL)*. It is defined by Ozbaygin et al. (2017) and solved to optimality by a Branch-and-Price (B&P) algorithm for instances with up to 60 customers. All these instances have been solved by (Yuan, 2019; Yuan et al., 2021) which present better results for larger instances. Note that these two solution approaches can also be used to solve instances of the GVRPTW.

Sitek and Wikarek (2019) defined the *Capacitated VRP with Pick-up and Alternative Delivery (CVRPPAD)*. This paper considers multiple options for each parcel to be delivered. Lockers and post offices are modeled with limited capacities. Customer preferences are modeled with a penalty on the objective function if a non-desired option is used. No time windows are considered. Their method relies on a pre-processing phase done by constraint programming. A heuristic then groups parcels together before assigning them to a route.

Zhou et al. (2018) defined the *Multi-Depot Two-Echelon Vehicle Routing Problem with Delivery Options (MD-2EVRP-DO)*. In this paper, a parcel can either be delivered at the customer's home or to a selected pick-up facility. Shared locations are incapacitated and customer preferences are modeled through penalties in the objective function. Time windows

are not considered. This problem is solved with a multi-population genetic algorithm that embeds an ad-hoc local search.

Since the publication of the presentation of the VRPDO in Dumez et al. (2021a) and Tilk et al. (2020) the interest of the research community about the utilization of lockers for parcel delivery has increased, and a few paper were published on the topic.

Mancini and Gansterer (2021) defined the *Vehicle Routing Problem with Private and Shared Delivery Locations (VRPPSDL)*. The VRPPSDL seeks to deliver the parcels of each customer either at home (under time windows constraint) or through a locker (with compensation cost). In addition, lockers have a capacity expressed as the number of customer that can possibly be delivered here. A fix-and-optimize method was developed to solve the VRPPSDL with an arc-based MIP model of the problem. Instances were solved with up to 75 customers and 5 lockers with a total capacity of 80.

Grabenschweiger et al. (2021) defined the *Vehicle Routing Problem with Heterogeneous Locker Boxes (VRPHLB)*. This problem assumes that parcels to be delivered and locker boxes fall under size categories such that the demand of each customer can fit inside one or more locker boxes. The VRPHLB seeks to deliver the parcels of each customer either at home (under time windows constraint) or through a locker (with compensation cost). But lockers have a limited size expressed as a number of boxes of each size. Their method integrates an ALNS heuristic to decide on the routing and bin-packing tailored heuristics to decide on the assignment of parcels to lockers or home delivery. They solved the instances of Mancini and Gansterer (2021), and instances with up to 75 customers, 5 locker stations, and 3 sizes of parcels matching with the 3 sizes of locker boxes. They showed that the cost of taking into account the packing of parcels into locker boxes is only of 1.3%.

Buzzega and Novellani (2021) defined the *Vehicle Routing Problem with Lockers and Time Windows (VRPLTW)*. This problem considers that locker are subject to fixed opening cost and capacity expressed as the number of customer that can be delivered here. Multiple mathematical formulations are detailed, and a branch-and-cut is developed. Instances with up 100 customers and 10 lockers of size 5 were proposed, but only instances with up to 40 customers and 4 lockers were consistently solved.

1.3.5 Synthesis

Table 1.1 summarizes the presented vehicles routing problems with resources synchronization. The first columns groups studies according to the class of the problem solved. Column 2 is

the reference to the study, Column 3 is the name of the solved problem, Column 4 is the name of the solution method. Column 5 indicates whether the considered synchronized resources are renewable, contrary to synchronized resources used during the full time period. The last column briefly describes the core idea of the solution methods to manage the synchronized resource constraints.

Tables 1.2 presents a non-exhaustive lists of scientific contribution related to the VRPDO, in the sense that they are vehicle routing problems with time windows in which not all vertex have to be visited. The first columns groups studies according to the class of the problem solved. Column 2 is the reference to the study, Column 3 is the name of the solved problem, Column 4 is the name of the solution method. The last column briefly describes the core idea of the solution methods to manage the alternative options.

problem	article	sub-problem	method	time dep.	treatment of the resource
VRP	Hempsch and Irnich (2008)	MDVRPTW	LS	✓	giant tour
	Grangier et al. (2017)	VRPCD-RC	LNS-SPM	✓	branch and check
	Froger et al. (2017)	EVRP-NL-C	2 phase : ILS, MIP	✓	branch and check + benders decomposition
	Mancini and Gansterer (2021)	VRPPSDL	ILS x LNS		repair with MIP
	Grabenschweiger et al. (2021)	VRPLB	ALNS-SPP		bin-packing for the lockers
	Buzzega and Novellani (2021)	VRPLTW	B&C		flow-based model
MVTPP	Gendreau et al. (2016)	MVTPP-PIC	B&P		
TOPTW	Souffriau et al. (2013)	MC-TOP-MTW	GRILS		boolean variable for each pair (option,route) and penalty in the objective for the infeasability

Table 1.1 – Vehicle routing problems with synchronized resources

problem	article	sub-problem	method	treatment of the alternatives
GVRP	Bektaş et al. (2011)	GVRP	LNS then B&C	graph on the cluster and cuts to choose an option
	Moccia et al. (2012)	GVRPTW	tabou	change the option of the close clients
	Afsar et al. (2014)	GVRP-flex	ILS	alternate between the routes and the giant tour
	Yuan et al. (2020b)	GTSPTW	B&C	
	Yuan et al. (2021)	GVRPTW	B&P-LS	exact choice of the options by DP change the option of the adjacent clients
VRPMTW	Tricoire et al. (2010)	MuP-OPTW	VNS	LS on the routes then choice of the time window to use
	Souffriau et al. (2013)	MC-TOP-MTW	GRILS	penalized temporal infeasibility
	Belhaiza et al. (2014)	VRPMTW	HVNTS	temporal infeasibility (backward time slack)
	Hoogeboom et al. (2020)	VRPMTW	AVNS	dynamic forward time interval in local search
TOPTW	Tricoire et al. (2010)	MuP-OPTW	VNS	perturbations
	Souffriau et al. (2013)	MC-TOP-MTW	GRILS	restarts et perturbations
	Hu and Lim (2014)	TOPTW	SA+SPP	perturbations and SPP
TPP	Gendreau et al. (2016)	MVTPP-PIC	B&P	
	Bernardino and Paias (2018)	UTPP	GA	product $\rightarrow$ vendor association in the genes
CTP	Allahyari et al. (2015)	MDCTVRP	GRASP+ILS+SA	specific neighborhoods; perturbations
	Takada et al. (2015)	m-CTP	ILS	dynamic penalty; specific neighborhoods; DP
	Vargas et al. (2015)	CTP	ALNS	spited giant tour via SPPRC
	Kammoun et al. (2017)	m-CTP	VNS	perturbations
other VRP	Reyes et al. (2017)	VRPRDL	LNS	DP; perturbation
	Zhou et al. (2018)	MD-2EVRP-DO	GA	options are coded into the genes
	Sitek and Wikarek (2019)	CVRPPAD	fix&optimize	heuristic choice
	Mancini and Gansterer (2021)	VRPPSDL	ILS x LNS	customer $\rightarrow$ location association by ILS
	Grabenschweiger et al. (2021)	VRPLB	ALNS	phase 1 : home only, phase 2 with options
	Buzzega and Novellani (2021)	VRPLTW	B&C	valid inequalities

Table 1.2 – Vehicle routing problems related to the VRPDO

1.4   Large Neighborhood Search for the VRPDO

Algorithm 1 presents the main steps of the Large Neighborhood Search (LNS) heuristic. The main loop of the iterative process is from lines (2) to (11). In Line (4) to select a recreate operator σ^+ from a set Σ^+ of operators. The same process is used in Line (5), a ruin operator σ^- is randomly selected from a set Σ^- of operators. Each operator has a given constant probability of being selected. In Line (6), the size Φ of the destruction is randomly chosen in a given interval $[\delta, \Delta]$. The destruction size is the percentage of customers to be removed from the current solution.

The selected operators are applied to the current solution s' in Line (7). First, $\Phi\%$ of the customers are removed from the solution with the chosen ruin operator σ^-. These customers are placed in the so-called request bank. Second, the customers from the request bank are inserted in the solution by the recreate operator σ^+. In Line (8), an acceptance criterion is used to decide whether the new solution becomes the current solution for the next LNS iteration. The acceptance criterion, and the penalization the customers in the request bank, is presented in Section 1.4.4.

The VRPDO has two lexicographic objectives: (1) to minimize the number of vehicles, and (2) to minimize the routing costs (described by the objective function 1.b). Similarly to Ropke and Pisinger (2006a), Algorithm 1 is run twice, with half of the time budget for each part. During the first phase, the number of vehicles is decreased by removing the smallest route from the solution each time a feasible solution is found (the customers are placed in the request bank). During the second phase, the routing costs are minimized, using the minimum number of vehicles found in a feasible solution obtained during the first phase. The configuration of our algorithm does not change between the two phases.

1.4.1   Ruin operators

Ruin operators use different rules to remove customers from the solutions. Upon removal, customers are placed in the request bank of the solution.

We divide ruin operators from the literature into two categories: *local ruin operators* and *large ruin operators*. A local ruin operator deletes customers so that even if few customers are deleted, it is likely that the solution will be improved on reconstruction. On the contrary, a large ruin operator requires that more customers be deleted. Indeed, with a large ruin operator, if too few customers are removed, it is likely that they will be re-inserted in the same position.

In the literature, large ruin operators are typically used with large destruction sizes. Whereas

---

Algorithm 1: LNS

1: s : initial solution
2: **while** the time budget is not reached **do**
3:     $s' \leftarrow s$
4:     randomly select a recreate operator $\sigma^+ \in \Sigma^+$
5:     randomly select a ruin operator $\sigma^- \in \Sigma^-$
6:     randomly select a destruction size $\Phi \in [\delta, \Delta]$
7:     $s' \leftarrow \sigma^+(\sigma^-(s', \Phi))$
8:     **if** s' meets the acceptance criterion **then**
9:         $s \leftarrow s'$
10:    **end if**
11: **end while**
12: **return** the best feasible solution found

---

the local ruin operators are from papers favoring speed of reconstruction with low destruction size but a larger number of iterations.

In the following, operators identified with [S] are the ones selected during the configuration study presented in Section 1.4.3. On the contrary, the discarded ones are indicated with [D].

Local ruin operators

The local ruin operators implemented from the literature are:

[S] *Distance-related removal* (Ropke and Pisinger, 2006b) : removes customers that are close to each other with respect to the Euclidean distance.

[D] *Node neighborhood removal* (Demir et al., 2012): removes customers that are close to each other with respect to the infinity norm.

[D] *Proximity removal* (Prescott-Gagnon et al., 2009): removes customers that are close both from a spatial and a temporal point of view, according to a parameterless formula. This is an extension of Shaw removal (Shaw, 1998).

[S] *(Split) String removal* (Christiaens and Vanden Berghe, 2020): removes sequences of customers in the routes of the current solution, either conserving, or not, a sub-string in the middle.

Large ruin operators

The large ruin operators implemented from the literature are:

37

[S] *Random removal* (Ropke and Pisinger, 2006b): randomly removes customers.

[D] *Demand-related removal* (Demir et al., 2012): removes customers that have demands of similar size.

[S] *Time-related removal* (Pisinger and Ropke, 2007): removes customers that are delivered at approximately the same time.

[S] *Zone removal* (Demir et al., 2012): randomly removes customers into predefined fixed rectangular zones.

[S] *Cluster removal* (Pisinger and Ropke, 2007): removes customers that are served by the same route in the current solution. A route is randomly selected and the Kruskal's algorithm is run on the arcs of this route until two clusters remain. All the customers in one of them, randomly chosen, are removed.

[S] *Route removal* (Nagata and Bräysy, 2009): removes all the customers of a route.

[D] *Distance worst removal* (Ropke and Pisinger, 2006a): iteratively removes the customer with the highest individual service cost. The individual service cost of a customer is the cost of the arcs that enter and exit the location where the given customer is served in the current solution, minus the cost of going directly from the previous location on the route to the next one. If it is a shared location, this cost is divided by the number of customers served at this location on the same route.

[D] *Time worst removal* (Demir et al., 2012): removes the customers that cause the largest time loss.

[D] *Neighborhood removal* (Demir et al., 2012): first, the cost of each route divided by the number of customers served by this route is computed. The customers are then removed sequentially by decreasing order of the difference between their individual service cost and the average service cost of their route.

[D] *Node-pair history removal* (Pisinger and Ropke, 2007): memorizes the cost of the best solution that uses each arc. The operator removes the nodes that are reached via arcs with the largest score.

[S] *Historical knowledge node removal* (Demir et al., 2012): this history removal memorizes the lowest individual service cost of each customer. The operator removes the customers with the largest difference between their current individual service cost and their lowest individual service cost. It can be see as a history-biased worst removal.

Ruin operators for the VRPTW can be adapted to the VRPDO in two ways: option-based or customer-based. An option-based operator only takes in account the options that are currently visited to serve the customers. A customer-based operator takes all the options into account. Let us describe an example with *distance-related removal*. The distance between two customers in *distance option-based related removal* is the distance between the options that are currently visited to serve these customers. On the contrary, in *distance customer-based related removal*, it is the minimal distance between any two options of these customers. That is to say, the option-based version will remove customers that are currently served in close locations and the customer-based one will delete customers that are potentially served in close locations.

VRPDO specific ruin operators

The new ruin operators specifically developed for the VRPDO are:

[D] *Preference-oriented random removal*: randomly selects customers and deletes them with a probability based on the preference level of the options currently visited to serve them. The probability of deleting a selected customer, currently served with option o, is $(1/P+1-p_o)^3$. When there are three preference levels, the probability of deleting a customer served with an option of level 3, 2 and 1 is 1.0, 0.125 and 0.04, respectively.

[S] *SDL-oriented random removal* (Shared Delivery Location-oriented random removal): randomly selects customers in the solution. If the selected customer is delivered at an individual location, the probability of being deleted is only 10%. Otherwise, if customers are delivered at a shared location, they are always deleted.

[D] *Random SDL removal*: randomly selects a shared delivery location and removes all the customers delivered at this location.

[D] *SDL-related removal*: selects a shared delivery location and randomly deletes customers that have an option at this location.

[D] *SDL-worst removal*: a *distance worst removal* where the detour cost is fully assigned to all the customers served at the shared delivery locations. The detour cost is not divided by the number of customers served at this location.

All these ad-hoc ruin operators are large ruin operators. With the exception of the *SDL related removal*, they are all option-based.

1.4.2 Recreate operators

Most recreate operators follow the best insertion principle: any given customer is inserted at the position that minimizes the increase of routing costs. To compute the best insertion of customer c in route r, we try to insert all customer options in all positions of route r. Only feasible insertions are performed by the algorithm. Hence, a solution always satisfies the vehicle capacity constraints, time windows, shared location capacities, and service level constraints. The only form of infeasibility considered in LNS is the fact that not all the customers are served, i.e the request bank can be non-empty.

The *forward time slacks* (Savelsbergh, 1992) of all routes are stored in order to evaluate insertions in constant time with respect to time windows. The usage of each shared location and of each preference level is stored. Hence, testing the validity of insertions with respect to synchronized resources is done in constant time. When an insertion is performed, the forward time slacks of the corresponding route are updated in linear time with respect to the length of the route. The update of capacity usage is done in constant time.

Recreate operators from the literature

The LNS recreate operators from the literature can be divided into two categories: *list heuristics* and others. Most of the traditional LNS repair operators rely on the evaluation of the insertion cost of each customer in the request bank at each position in the solution. Typically, the insertion of each customer, in the request bank, into each route is evaluated once at the beginning. After each modification, the insertion of the remaining customers in the modified route is then re-evaluated. In their recent revisited version of LNS, Christiaens and Vanden Berghe (2020) use exclusively list heuristics. For each insertions, list heuristics first select a customer in the request bank according to some criterion and second, evaluate the insertion cost of the selected customer in the solution routes. Accordingly, list heuristics are often more naive but faster and easier to implement.

The list heuristics implemented from the literature are:

[S] *Random order best insertions* (Christiaens and Vanden Berghe, 2020): sequentially inserts the customers in the request bank at their best insertion position in a random order.

[D] *Oldest first best insertions* (Christiaens and Vanden Berghe, 2020): sequentially inserts the customers in the request bank at their best insertion position in non-increasing order of the number of iterations since the last time the considered customer was served.

[S] *Largest first best insertions* (Christiaens and Vanden Berghe, 2020): sequentially inserts the customers in the request bank at their best insertion position in non-increasing order of their demand.

[D] *Farthest first best insertions* (Christiaens and Vanden Berghe, 2020): sequentially inserts the customers in the request bank at their best insertion position in non-increasing order of distance to the depot.

[D] *Closest first best insertions* (Christiaens and Vanden Berghe, 2020): sequentially inserts the customers in the request bank at their best insertion position in increasing order of their distance to the depot.

The other operators implemented from the literature are:

[D] *Best temporal insertions* (Demir et al., 2012): inserts the customers so that the loss of time is minimal. The loss of time is defined as the waiting time at the inserted option plus the waiting time at the next option on the route. Customers can be processed either in random order or by decreasing order of demand, respectively.

[D] *Greedy best insertion* (Ropke and Pisinger, 2006a): iteratively computes the cheapest insertion for each customer and inserts the customers that have the lowest insertion cost.

[S] *k-regret* (Ropke and Pisinger, 2006a): iteratively computes the best insertion cost on each route for each customer and inserts the one that has the largest difference between its best insertion cost and next $(k - 1)$ route's best insertion costs.

[S] *Ejection search* (Nagata and Bräysy, 2009): first, all the customers in the request bank are placed in a FIFO structure. The customers from this structure are inserted into the solution by allowing some customers from the solution to be removed and put in the queue. As in Curtois et al. (2018) the procedure is heuristically sped up. First, to insert one customer, at most two customers can be removed from the solution. Second, insertions are tested with an increasing number of removed customers. If a feasible insertion is found, insertions with more removals will not be tested. Third, when customers must be removed to insert a customer, the insertion that removes the customers with the smallest score is chosen. The score of a customer is the number of times where no feasible insertion was found for this customer during all the calls to *ejection search*. Finally, the number of iterations of *ejection search*, at each call, is limited to five times the initial size of the request bank.

VRPDO specific recreate operators

The operators specifically developed for the VRPDO are:

[S] *Preferred best insertion*: this operator is an adaption of the classical best insertion operator. The only difference is that the insertion costs are modified in a lexicographical manner: the cost of all insertion positions with all delivery options are calculated, then these insertions costs are lexicographically sorted by considering the preference levels p_o first and the insertion costs second.

[D] *Preference regret*: the best insertion is computed for each customer and for each preference level. Let C_p^i be the cost of the best insertion of a customer i with an option of level p or lower. The *preference regret* score of customer i is $\sum_{p=1}^{P-1}(C_P^i - C_p^i)$. Customers are always inserted at their cheapest position and the customer with the largest regret is inserted first.

[D] *Normalized best insertion*: the normalized insertion cost of a customer at a given location is the cost of the insertion divided by the capacity of the location. *Normalized best insertion* is a *greedy best insertion* that uses an normalized insertion cost to select the insertion possibility for each customer and select the first customer to insert.

[S] *SDL-regret* (Shared Delivery Location regret): customers are inserted at their cheapest insertion in non-increasing order of regret. In this version, the regret of a customer is the difference between the insertion cost when all options are allowed and its insertion cost when only individual locations are authorized. For example, let us consider the partial solution represented in Figure 1.8 (same instance as in Figure 1.3). Customers 5 and 6 are not served. Both can be delivered at locker II, like customer 4. But this locker only has a capacity of two, as in Figure 1.5. For both customers 5 and 6, the cheapest insertion is in this locker, with a cost of 0. Figures 1.9 and 1.10 show the cheapest insertion of customers 5 and 6 without considering shared locations. Hence, SDL-regret for customer 6 is higher than that of customer 5. In this case, the *SDL-regret* will select customer 6 first and insert him/her in the locker.

1.4.3 Operator selection

As described in sections 1.4.1 and 1.4.2, a number of operators have been implemented. Because it does not seem useful to keep them all, we searched for a configuration of LNS with fewer operators. In this section, we define a *configuration* as a subset of ruin operators and a subset of recreate operators.

A statistical study was performed to choose a configuration from the 20 ruin operators and the 15 recreate operators that were implemented. Tests were performed on a representative set of
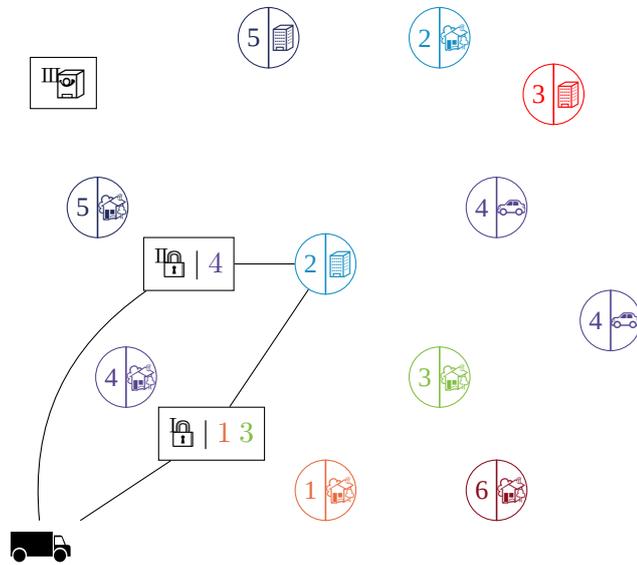
Figure 1.8 – Example of a partial solution
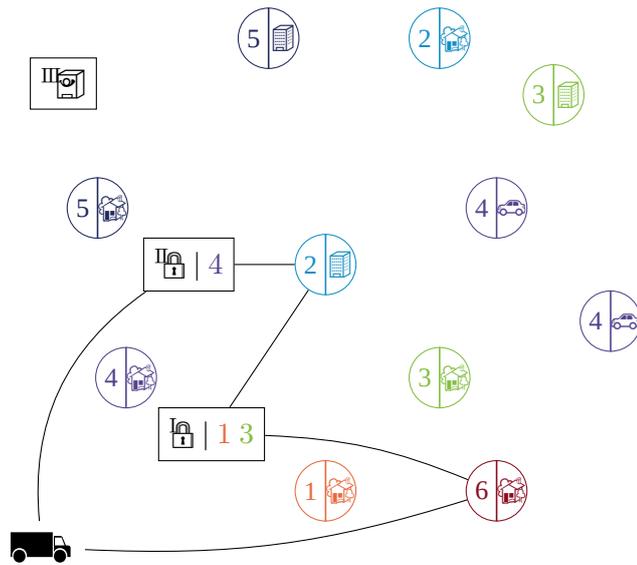
Figure 1.9 – Insertion of the brown (6) customer without lockers

randomly generated VRPDO instances. To determine whether one configuration is significantly different from another, the *Wilcoxon pairwise test* (Wilcoxon, 1992) was used with a threshold of 5%. This test compares two populations of results. In our case, it compares the results obtained by two configurations on each instance of a set of VRPDO instances. This methodology is inspired from Stützle (2018). Note that the automatic configuration package IRACE (López-Ibáñez et al., 2016) was also used, but it did not converge after several days of computation.

43
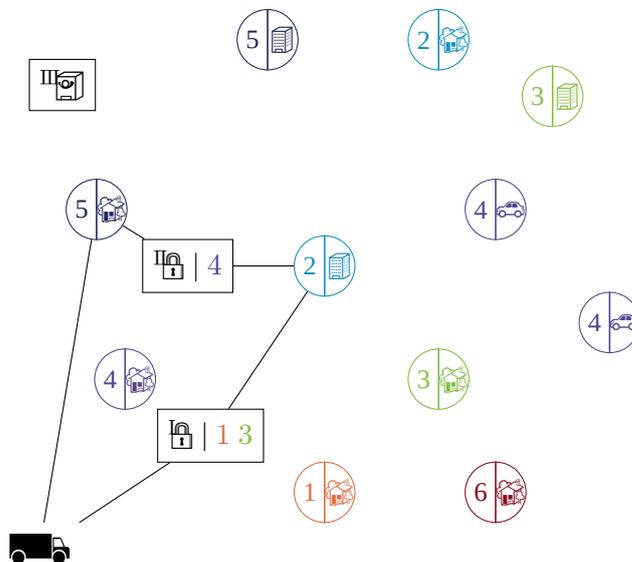
Figure 1.10 – Insertion of the purple (5) customer without lockers

Let us define a class of operators as a group of operators that have similar purposes. First, the ruin operators are split into 4 classes: *random removals, related removals, worst removals,* and *history removals*. Second, the recreate operators are split into 4 classes: *list heuristics, time best insertion heuristics, regret heuristics,* and *ejection search*. This classification is detailed in Tables 1.3 and 1.4.

This statistical study is decomposed in two phases: (1) a study of the impact of each class of operators; (2) a study of the impact of each individual operator.

To study classes of operators, the reference configurations are: the full configuration with all the operators, and a minimal configuration with as few operators as possible. All the operators of each class are removed from the full configuration and added to the minimal configuration. All these "sub-configurations" were tested on all instances of the test set and compared. This first phase determines whether the considered operators are redundant with other operators of the full configuration, and whether they improve the results of the minimal configuration.

Based on the previous results, we build an intermediate configuration. The operators from this configuration are changed one by one. If a given operator was used, then it is deactivated, otherwise it is added to the configuration. All these alternative configurations are compared with the intermediate configuration. It determines if the assessed operator significantly improves the results, or if it is redundant with used operators.

The results of the operators are summarized in Tables 1.3 and 1.4. Operators marked with [++] are considered essential. Operators marked with [+] slightly improve the results. Operators marked

with - are useless or redundant with already kept operators, and adding them does not improve the results. The selected operators are those rated ++ and +.

	Option-based operators	Customer-based operators
Random removals	- preference-oriented random removal + SDL-oriented random removal - random SDL removal	+ random option removal
Related removals	- distance-related option removal - node neighborhood removal - time-related option removal ++ (split) string removal ++ cluster removal + route removal	+ distance-related customer removal + zone removal + time-related customer removal - proximity customer removal - demand-related removal - SDL-related removal
Worst removals	- distance worst removal - time worst removal - neighborhood removal - SDL worst removal	
History removals	- node-pair history removal	++ historical knowledge node removal

Table 1.3 – Overview of configuration experiments for ruin operators

1.4.4 Acceptance Criterion

In line 8 of Algorithm 1, the acceptance criterion determines whether the newly generated solution should be accepted as the current solution at the next iteration.

Ropke and Pisinger (2006b) used the Metropolis criterion from Simulated Annealing (Kirkpatrick et al., 1983). To deal with partial solutions, a modified cost was proposed in Pisinger and Ropke (2007). The modified cost of Equation (1.1) penalizes the unserved customers with factor β. In this formula, B is the request bank of the current solution, $cost$ is its routing cost (described by 1.b) and N is the set of customers.

$$\text{modified cost} \;=\; \text{cost} \times \left(1 + \beta . \frac{|B|}{|N|} \right) \tag{1.1}$$

Santini et al. (2018) conducted a comprehensive study on the acceptance criteria for the LNS metaheuristic. In their conclusion about the CVRP, they advocated for the record-to-record

	VRPTW operators	VRPDO operators
List heuristics	⁺⁺ random order best insertion ⁻ oldest first best insertion ⁺⁺ largest first best insertion ⁻ farthest first best insertion ⁻ closest first best insertion	⁻ normalized best insertion ⁺ preferred best insertion
Time best insertions	⁻ time best insertion	
Regrets	⁻ greedy best insertion ⁺⁺ 2-regret ⁻ 3-regret ⁻ 4-regret	⁺ SDL-regret ⁻ preference regret
Ejection search	⁺ ejection search	

Table 1.4 – Overview of configuration experiments for recreate operators

criterion (Dueck, 1993). With this criterion, the solution is accepted if its modified cost is less than $T\%$ larger than the modified cost of the best known solution. Furthermore, they propose decreasing the acceptance threshold T during the algorithm. T decreases linearly between its initial value at the beginning and 0, when the time limit is reached. They conclude that the best values of T and β depend of the type and size of instance.

Our implementation uses the record-to-record criterion with modified cost (1.1). To avoid tuning parameters and get a reliable acceptance criterion, a simple adaptive procedure is proposed.

Our experiments empirically show that LNS performs well if the ratio of accepted solutions is between 4% and 14%. T and β are changed so as to maintain the ratio of accepted solutions in this target. The ratio of accepted solutions is periodically evaluated. If the ratio of accepted solutions is less than 4%, T is multiplied by 1.5 and β is divided by 1.5. On the contrary, if it is larger than 14%, T is divided by 1.5 and β is multiplied by 1.5.

1.4.5 Parameters tunning

In our LNS, the probability of selecting each operator is constant throughout the algorithm. Indeed, Turkeš et al. (2020) reviewed the LNS literature and conclude that the adaptive feature has, at best, a very little positive impact. This observation was confirmed for our case by preliminary experiments.

In the proposed implementation, all ruin operators are equiprobable. The probability of selecting each recreate operator is inversely proportional to its average running time. Furthermore, Christiaens and Vanden Berghe (2020) propose to perform a huge number of small and fast iterations in LNS in order to compensate the lack of local search in this metaheuristic. We added this functionality as a special case: if a list heuristic is selected, there is a high probability ϕ that the destruction size will be small (between δ_{mini} and Δ_{mini} percent of the customer) and that the ruin operator will be local, i.e *string removal*, *split string removal* or *distance-related customer removal*. The probability of each operator, δ, Δ, δ_{mini}, Δ_{mini} and ϕ was tuned according to the recommendations of IRACE (López-Ibáñez et al., 2016).

Christiaens and Vanden Berghe (2020) introduced the *"blink"* principle for recreate operators. It randomly ignores certain insertions with a given probability during the computation of the best insertion. In the proposed implementation, this feature did not prove to have a significant impact. It introduces diversification through randomization. Nevertheless, this principle has been applied to the ruin operators; for each removal evaluated there is a given probability of simply ignoring it.

To summarize, on the one hand, the list heuristics and the small destructions favor a high number of iterations. On the other hand, using regret heuristics and ejection search tends to reduce the number of iterations. As observed by Christiaens and Vanden Berghe (2020), the small destruction and list heuristic can compensate for a lack of local search. Furthermore, the numerous iterations coupled with blink provide a good exploration of the search space in the CVRP. To deal with time windows, we observe that it is worthwhile to perform larger destruction and to take some time to anticipate constraint violation, like Ropke and Pisinger (2006a) and Demir et al. (2012).

Our LNS always works in two phases: In the first phase, it reduces the number of routes, and in the second phase, it minimizes the routing cost using the established number of routes. The first phase is either stopped when half of the time budget has been used, or sooner when a feasible solution with the given number of routes has been found.

To conclude this section, we indicate the values of the parameters used for LNS (tuned with the help of IRACE) :

— The probability of selecting each ruin operator is the same for all operators. The probability to blink a deletion possibility is 30%.

— The probability of selecting a recreate operator is inversely proportional to its running time. The probability of selecting list heuristics (*random order best insertion, largest first best insertion*) is 0.4. The probability of selecting the other recreate operator (*2-regret*,

ejection search, *SDL-regret* and *preferred best insertion*) is 0.05.

— The destruction size and removal operator selection rule is different for the list heuristics and the other recreate operators. In general the destruction size is between $\delta = 10\%$ and $\Delta = 20\%$ of the number of customers. For the list heuristic there is a 30% probability of performing a classical destruction (using any ruin operator) and a $\phi = 70\%$ probability of performing a small, local destruction. That is to say, only between $\delta_{\text{small}} = 1\%$ and $\Delta_{\text{small}} = 10\%$ of customers are removed, and a local removal operator (*distance-related removal, (split) string removal*) is used. For large instances, the destruction sizes are limited to 20 and 80 customers respectively.

— The initial values of the record-to-record acceptance criteria are $T = 0.18$ and $\beta = 9$. These values are adjusted every 4500 iterations by a factor of 1.5.

1.5   LNS with small destruction

After many rounds of tests the LNS presented in Section 1.4 was improved. In this section we will focus on a better orchestration of small and large destruction phases. This LNS framework is used in Dumez et al. (2021c).

The utilization of the LNS framework presented in the previous section mostly relies on the ideas presented in Ropke and Pisinger (2006a), Demir et al. (2012), and Christiaens and Vanden Berghe (2020). The modification of the framework presented in this section are inspired from Hemmelmayr et al. (2012). They proposed an LNS heuristic for the two-echelons vehicles routing problem. In their method, most of the iterations focuses on the routing at the second echelon, and the first echelon is modified only if the solution was not improved during the last iterations.

1.5.1   Algorithm

We can now present the synopsis of the proposed LNS heuristic in Algorithm 2: *Small and Large Neighborhood Search (SLNS)*. In this pseudo code, s is the current solution, s^* is the best-found solution, and s' is a copy of the current solution to be modified. Moreover, Σ^+ is the set of repair operators, Σ^- the set of destroy operators, and $\Sigma^-|_{\text{local}} \subset \Sigma^-$ is the set of local destroy operators (see Section 1.4.3). The variable $iter$ counts the number of iterations since the last new best solution was found or the last large destroy was performed.

The algorithm is initialized by setting $iter$ to zero (Line (2)). The main loop is given by the Lines (3) to (24). In each iteration, either a local or large destruction operator is selected depending on the value of $iter$ and the input parameter ω: If $iter < \omega$, the iteration counter is increased, the current solution is copied and a small destruction using a local destroy operator in $\sigma^- \in \Sigma^-|_{\text{local}}$ with destruction size in $[\delta_{\text{small}}, \Delta_{\text{small}}]$ is performed (Lines (6) to (9)). Otherwise, the counter $iter$ is reset, the best-found solution is copied and a large destruction is performed, i.e., the destroy operator is randomly selected in Σ^- and the destruction size is chosen at random in $[\delta_{\text{large}}, \Delta_{\text{large}}]$ (Lines (11) to (14)). A repair operator is randomly selected in $\sigma^+ \in \Sigma^+$ (Line (4)). Then, the combination of the selected operators, σ^- and σ^+, is applied to solution s' in Line (16).

The new solution becomes the current solution in Line (17) if its modified cost is smaller than the current or a large destruction was performed. In Line (20), the best-found solution may be updated and the counter $iter$ is reset accordingly.

Algorithm 2: SLNS

1: s : initial solution
2: $iter = 0$
3: **while** the time budget is not reached **do**
4:      randomly select a recreate operator $\sigma^+ \in \Sigma^+$
5:      **if** $iter < \omega$ **then**
6:          $s' \leftarrow s$
7:          $iter \leftarrow iter + 1$
8:          randomly select a local ruin operator $\sigma^- \in \Sigma^-|_{\text{local}}$
9:          randomly select a destruction size $\Phi \in [\delta_{\text{small}}, \Delta_{\text{small}}]$
10:      **else**
11:          $s' \leftarrow s^*$
12:          $iter = 0$
13:          randomly select a ruin operator $\sigma^- \in \Sigma^-$
14:          randomly select a destruction size $\Phi \in [\delta_{\text{large}}, \Delta_{\text{large}}]$
15:      **end if**
16:      $s' \leftarrow \sigma^+(\sigma^-(s', \Phi))$
17:      **if** $f'(s') < f'(s)$ or $iter = 0$ **then**
18:          $s \leftarrow s'$
19:      **end if**
20:      **if** $f(s') < f(s^*)$ **and** $B_{s'} = \emptyset$ **then**
21:          $s^* \leftarrow s'$
22:          $iter \leftarrow 0$
23:      **end if**
24: **end while**
25: **return** s^*

In many applications, the strength of LNS relies on its speed. Pisinger and Ropke (2007) reports that their ALNS takes 146 seconds, on average, for performing 50,000 iterations on the Solomon instances for the VRPTW with 100 customers. With the same time budget, on the same instances, SLNS performs 7.4 million iterations on average, i.e., approximately 150 times more iterations with a CPU that is only 1.91 times faster per thread according to (PassMark-Software, 2020).

Like SISR (Slack Induction by String Removals) proposed by Christiaens and Vanden Berghe (2020), SLNS perform a tremendous number of fast iterations to locally improve the solutions. But experiments showed that SISR lacks of exploration to be competitive on the VRPDO. On the contrary the previously presented LNS is a lot more focused on diversification, similarly to the ALNS of Ropke and Pisinger (2006a). But such configurations of LNS significantly suffer from increase in size of instances.

Algorithm 2 uses both the intensification capabilities of SISR and the explorations ones of Algorithm 1. Thus, SLNS is both efficient and able to reach high quality solutions, allowing it to be competitive on large VRPDO instances as we will see in Section 1.6.

1.5.2   Parameters tunning

In SLNS, the acceptance criterion is a threshold acceptance (Dueck and Scheuer, 1990) with a constant temperature of 0: if the new solution improves the current one, it is accepted. In addition, solutions produced by a large destruction are always accepted. In order to manage solutions with non-empty request bank we still compare the solutions with the modified cost 1.1. We set a constant penalization factor $\beta = 20$ in Equation 1.1.

SLNS use the same operators as LNS with the same constant probability (see Section 1.4.5). Moreover, we set the size of the small destruction interval to $[\delta_{\text{small}}, \Delta_{\text{small}}] = [0.01|N|, 0.1|N|]$ and the size of the large destruction interval to $[\delta_{\text{large}}, \Delta_{\text{large}}] = [0.1|N|, 0.3|N|]$. For large instances, the destruction sizes are limited to 20 and 80 customers respectively. Finally, The number of iterations between two large destructions is set to $\omega = 10|N|^{1.5}$. For example, $\omega =$10 000 iterations with 100 customers and 80 000 with 400 customers.

1.6   Experiments on LNS

In this section, we compare the two proposed solution frameworks on the VRPTW benchmark instances and on the newly generated VRPDO instances.

The methods are coded in C++ and is compiled with g++ 5.4.0. Experiments on the VRPDO instances were performed using Linux, Ubuntu 16.04 LTS, running on an Intel Xeon X5650 @ 2.57 GHz. Experiments on the VRPTW instances were performed using Linux, Ubuntu 20.04.2 LTS, running on an Intel Xeon Gold 6230 @ 2.10GHz.

1.6.1   Results on the vehicle routing problem with time windows

The LNS (Algorithm 1) and SLNS (Algorithm 2) were evaluated on 176 benchmark instances of the VRPTW, proposed by Solomon and Desrosiers (1988) (100 customers) and Gehring and Homberger (1999) (200 and 400 customers). Instances of type R are composed of randomly located customers. Instances of type C are composed of clustered customers. Instances of type RC are a mix of random and clustered customers. Each type of instance includes two groups, in which routes visit a few customers and many customers, respectively.

Tables 1.5, 1.6 and 1.7 present the results for instances with 100, 200 and 400 customers, respectively. In each table, the results found by LNS and SLNS are compared with those obtained by the ALNS of Pisinger and Ropke (2007) and by the state of the art Hybrid Genetic Algorithm with Adaptive Diversity Management (HGSADC) of Vidal et al. (2013). The parameters of our methods, tuned for the VRPDO, were not changed for the VRPTW. Detailed results are available in Appendix A.2.

Each line represents the average result for one type of instance. Instance groups are listed in column 1. Columns 2, 3 represent the average number of routes in the solution of the ALNS and the average cost, over all instances of each group (only the best result out of 5 runs is considered). Columns 4, 5, and 6, 7 and 8, 9 give the same information for HGSADC, LNS, and SLNS respectively. The last lines give the total over all instances for each algorithm considered, and the time budget allocated to each algorithm, respectively.

Figure 1.11 shows the relative gap, in percentage, between the best known solutions and the best solutions found by LNS and SLNS out of 5 runs. For some instances, solutions with the optimal number of vehicles were not found during the first phase of LNS, thus, solutions may have a travel distance lower than the optimal one with the lowest number of vehicles.

These experiments show that, although it has been tailored for the VRPDO, LNS and SLNS

competes with state-of-the-art algorithms on instances with up to 200 customers. On larger instances SLNS remains fairly close to optimal solution, but the performances of LNS severely deteriorates. Indeed, in this chapter we solely focus on non-hybrid LNS, while the regular resolution of a set partitioning problem to recombine routes greatly improve the performances of LNS on the VRPTW. We refer the interested reader to the results published in Dumez et al. (2021a).

Instance	ALNS Øveh	ALNS Øcost	HGSADC Øveh	HGSADC Øcost	LNS Øveh	LNS Øcost	SLNS Øveh	SLNS Øcost
C1	10.00	828.38	10.00	828.38	10.00	828.38	10.00	828.38
C2	3.00	589.86	3.00	589.86	3.00	589.86	3.00	589.86
R1	11.92	1 212.39	11.92	1 210.69	11.92	1 221.97	12.00	1 209.04
R2	2.73	957.72	2.73	951.51	2.73	957.19	2.73	957.41
RC1	11.50	1 385.78	11.50	1 384.17	11.63	1 374.77	11.75	1 367.10
RC2	3.25	1 123.49	3.25	1 119.24	3.25	1 131.04	3.25	1 131.78
Total	405	57 332	405	57 196	406	57 413	408	57 205
Time (*s*)	150		160		100			

Table 1.5 – Solomon VRPTW instances with 100 customers

Instance	ALNS Øveh	ALNS Øcost	HGSADC Øveh	HGSADC Øcost	LNS Øveh	LNS Øcost	SLNS Øveh	SLNS Øcost
C1	18.90	2 718.77	18.90	2 718.41	18.90	2 748.24	18.90	2 723.98
C2	6.00	1 831.59	6.00	1 831.59	6.00	1 834.29	6.00	1 831.76
R1	18.20	3 615.69	18.20	3 613.16	18.20	3 788.85	18.20	3 662.15
R2	4.00	2 937.67	4.00	2 929.41	4.00	2 962.08	4.00	2 960.04
RC1	18.00	3 192.56	18.00	3 180.48	18.00	3 338.69	18.00	3 221.54
RC2	4.30	2 559.32	4.30	2 536.20	4.30	2 580.04	4.30	2 581.79
Total	694	168 556	694	168 092	694	172 521	694	169 812
Time (s)	3 180		504		500			

Table 1.6 – Homberger VRPTW instances with 200 customers

Instance	ALNS		HGSADC		LNS		SLNS	
	Øveh	Øcost	Øveh	Øcost	Øveh	Øcost	Øveh	Øcost
C1	37.60	7 182.75	37.60	7 170.47	37.60	7 987.74	37.70	7 236.68
C2	11.90	3 874.58	11.60	3 952.95	11.80	4 031.95	11.90	3 922.44
R1	36.40	8 420.52	36.40	8 402.57	36.40	10 165.36	36.40	8 571.28
R2	8.00	6 213.48	8.00	6 152.92	8.00	6 317.79	8.00	6 308.28
RC1	36.00	7 940.65	36.00	7 907.14	36.00	9 317.89	36.10	8 072.79
RC2	8.60	5 269.09	8.50	5 215.21	8.60	5 370.30	8.60	5 354.11
Total	1 385	389 011	1 381	388 013	1 384	431 910	1 387	394 655
Time (s)	5 340		2 046		1 900			

Table 1.7 – Homberger VRPTW instances with 400 customers
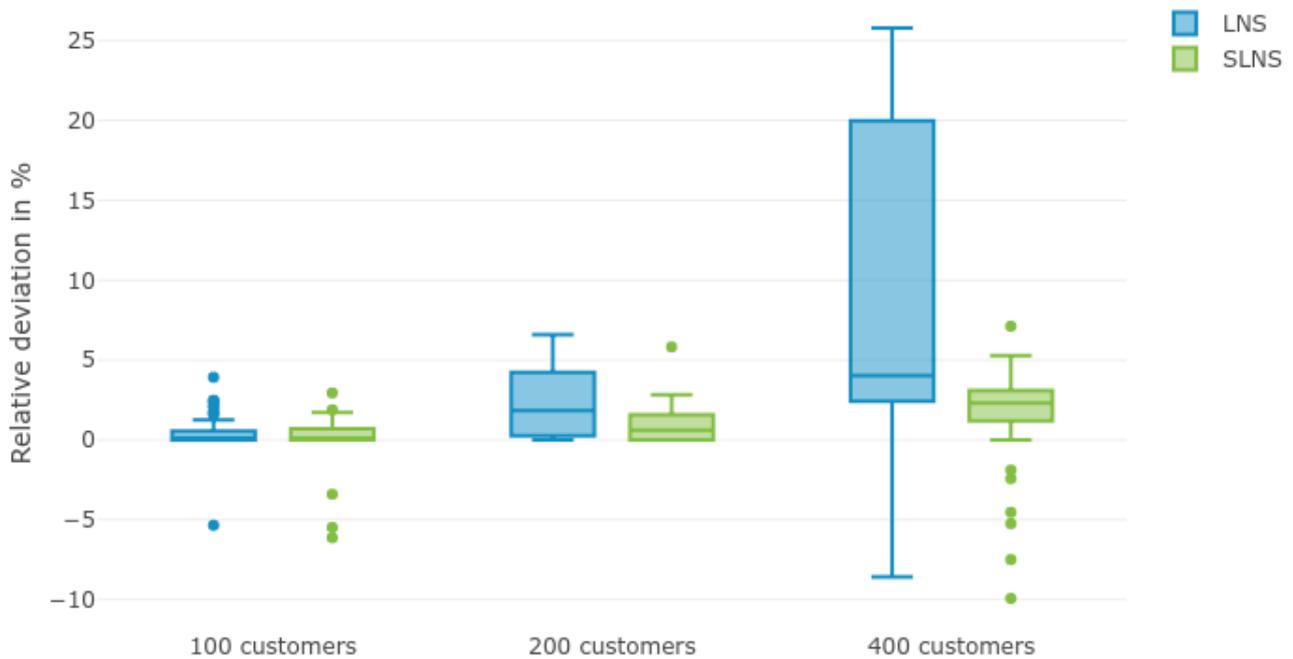
Figure 1.11 – Relative deviation between the results of the LNS or SLNS and the best known solution, on various VRPTW instances

1.6.2 Results on the vehicle routing problem with delivery options

VRPDO instances were randomly generated. Three types of instances were generated: U, V and UBC. For each type, instances with 50, 100, 200 and 400 customers were generated.

All the delivery locations were randomly generated in a 50×50 square. The depot is located at the bottom left-hand corner. Euclidean distances are considered. A unit of distance costs 1 and takes a unit of time to be crossed. A time horizon of 12 hours, i.e. 720 time units, is considered.

In the U and UBC instances, each customer has between 1 and 3 options, with an average of 2 options per customer. In the V instance, each customer has 1 or 2 options, with an average of 1.5 options per customer.

In the U and V instances, the capacities are tight, both for the vehicles and the lockers. A locker can accept between 3 and 5 parcels, and vehicle capacity is such that a route can serve around 10 customers. In the UBC instances (U with Big Capacity), the vehicle capacity is such that a route can serve around 25 customers. Furthermore, there are five times fewer lockers and their capacity is five times larger.

The time window of individual locations can be either: the morning ($[0; 360]$), the afternoon ($[360; 720]$), random in the morning (i.e $[a_i, b_i]$ such that $0 \leqslant a_i \leqslant 240$ and $b_i = a_i + 120$), random in the afternoon (i.e $[a_i, b_i]$ such that $360 \leqslant a_i \leqslant 600$ and $b_i = a_i + 120$) or random in the whole day (i.e $[a_i, b_i]$ such that $0 \leqslant a_i \leqslant 480$ and $b_i = a_i + 240$). The time window of a shared location can be either: random in the day (i.e $[a_i, b_i]$ such that $0 \leqslant a_i \leqslant 240$ and $b_i = a_i + 480$) or the full day ($[0; 720]$).

By default, all these tests were conducted with a service level of $80\% - 90\%$, i.e. at least 80% of the customers are served with their preferred option and at least 90% of the customers are served with an option of level 1 or 2.

The characteristics of instance classes are summarized in Table 1.8. For each size and each class, 10 instances were generated, leading to a total of 120 instances. All the instances are available upon request.

Table 1.9 summarize the results of different LNS heuristics on the VRPDO instances. To asses the performance the proposed configurations of LNS, we compare them to known configurations of the metaheuristic:

ALNS we configured our program as close as possible to the method presented in Pisinger and Ropke (2007) for the VRPTW. With our implementation, the average relative deviation between the best solutions found out of five runs and the best known solutions of the VRPTW is of 1.31% on the Solomon instances with 100 customers. Pisinger and Ropke

Instance type	Capacity	Avg. number of options per customer	Time windows width	
			Individual locations	Shared Locations
U	medium	2	2 to 6 hours	8 to 12 hours
UBC	big	2	2 to 6 hours	8 to 12 hours
V	medium	1.5	2 to 6 hours	8 to 12 hours

Table 1.8 – Classes of VRPDO instances

(2007) report a deviation to the optimum of 0.25% on these instances, with 146 seconds of time budget.

SISR we configured our program as close as possible to the method presented in Christiaens and Vanden Berghe (2020) for the VRPTW. With our implementation, the average relative deviation between the best solutions found out of five runs and the best known solutions of the VRPTW is of 1.08% on the Solomon instances with 100 customers. Christiaens and Vanden Berghe (2020) do not report results on the 100 customers instances, but report a relative diaviation to the optimum on the 200 customers instances of 0.25% with 11 minutes of time budget.

LNS we configured our method as described in Section 1.4.

SLNS we configured our method as described in Section 1.5.

Each line represents a group of instances. The type of instance, the number of customers, the number of delivery locations and the number of options are specified in columns 1, 2, 3 and 4, respectively. For each configuration we depict the total number of vehicles and the total routing cost on all the considered instances. The best solution out of five runs is taken into account. The computing time is indicated by Column 13.

Overall the VRPDO instances, SLNS clearly provides the best results. On average, it improves the results of ALNS by 4.9%, those of SISR by 8.3%, and those of LNS by 10%. ALNS, SISR, and LNS are able to find acceptable solution on instances with 100 customers, with an average relative deviations to the solutions of SLNS of 1.8%, 0.6%, and 0.2% respectively. On the contrary, these LNS configurations are significantly outperformed on 400 customers instances, with a average relative deviations to the solutions of SLNS of 9.4%, 23.4%, and 28.5% respectively.

Type	$|N|$	$|L|$	$|O|$	ALNS		SISR		LNS		SLNS		Time(s)
				Σveh	Σcost	Σveh	Σcost	Σveh	Σcost	Σveh	Σcost	
U	100	120	200	105	6 627.91	105	6 592.01	105	6 605.10	105	**6 537.50**	300
	200	240	400	205	12 492.15	205	12 744.03	205	13 522.29	205	**11 853.02**	2000
	400	480	800	406	25 918.78	406	30 153.17	**405**	31 249.29	406	**22 917.71**	6000
V	100	96	150	104	7 174.38	104	7 033.73	104	**6 998.78**	104	7 072.73	300
	200	192	300	204	14 601.32	204	14 505.64	204	15 031.16	204	**13 677.11**	2000
	400	384	600	407	23 656.96	405	28 075.53	405	28 685.68	405	**22 614.49**	6000
UBC	100	104	200	40	3 673.21	40	3 636.98	40	3 598.91	40	**3 580.71**	300
	200	209	400	80	6 420.59	80	6 366.18	80	6 362.56	80	**6 077.53**	2000
	400	416	800	156	12 410.63	156	12 933.65	156	13 766.62	156	**11 242.28**	6000
Total				1707	112 975.93	1705	122 040.92	1704	125 820.40	1705	**105 573.08**	

Table 1.9 – Results of different methods on the VRPDO instances

1.7 Managerial insights on the VRPDO

In this section we present managerial insights on the VRPDO about the impact of the delivery options on the routing costs and on the quality of service. Note that these results are those published in Dumez et al. (2021a), obtained by the standard LNS combined with a SPP (presented in Chapter 2).

Economic impact of delivery options

To quantify the impact of delivery options, the solution of the VRPDO instances are compared to those of their VRPTW counterpart. To transform a VRPDO instance into a VRPTW instance, we consider only the home option of each customer. No preference level is taken into account. In the VRPDO instances, home delivery options are assumed to be the preferred individual locations. In the case of customers that only have a locker option, a random location is added as a home location.

The total number of vehicles and the routing cost for each instance class are depicted in Table 1.10.

Type	$|N|$	VRPTW Σveh	VRPTW Σcost	VRPDO Σveh	VRPDO Σcost	Gap (%)	% SDL	SDL fill rate
U	50	54	5 504.07	54	3 864.48	29.8	58%	67%
	100	105	8 662.22	105	6 502.99	24.9	58%	67%
	200	205	16 223.30	205	13 641.13	15.9	60%	67%
UBC	50	20	5 253.71	20	2 293.53	56.3	67%	71%
	100	40	7 052.06	40	3 608.39	48.8	68%	67 %
	200	80	10 681.93	80	6 384.29	40.2	66%	64%
V	50	54	5 192.10	54	3 742.59	27.9	57%	72%
	100	104	9 877.77	104	7 089.06	28.2	56%	79%
	200	205	18 945.38	205	14 781.23	22.0	54%	78%
Total		867	87 392.53	867	61 907.68	29.2		

Table 1.10 – Economic impact of delivery options

Each line represents the average results over the 10 instances of each group. The first column indicates the type of instance and the second column the number of customers. Columns 3 and 4 indicate the total number of vehicles and the sum of routing costs over the instances of

the group when only home delivery is considered. Columns 5 and 6 indicate the total number of vehicles and the total routing cost over the instances of the group when all options are considered. Column 7 is the relative savings on the routing costs obtained by using delivery options. Considering delivery options leads to a cost reduction of 29.2%, on average. Furthermore, on the UBC instances, with large lockers, the savings are even larger. The number of routes is not reduced, because it is determined by the binding vehicles' capacities.

The use of SDLs is detailed in the last two columns, which detail the percentage of customers delivered at SDLs and the average fill rate of SDLs, respectively. These results indicate that the value of the gap in column 7 is related to the use of the shared locations. It is noticeable that a larger part of customers are delivered at SDLs in UBC instances with big capacities. But the last column shows that SDLs are not saturated. This suggests that alternative individual locations play a non negligible role in the gap between the VRPTW and the VRPDO.

Additionally, looking at the actual service levels on a representative set of solutions, we observe that the percentage of customers served with their option of level 1 (resp. level 2) is very close to the thresholds of 80% and 90%. The service level constraints in the model are often binding.

Delivery options are compatible with tight time windows

We performed a sensitivity analysis by modifying the width of individual locations time windows. Considering a time window $[a_i, b_i]$ at location $i \in L$, the modified time window is still centered at time $(a_i + b_i)/2$ but its width is reduced by a factor α as shown in formula (1.2).

$$[a'_i, b'_i] = \left[\frac{a_i + b_i}{2} - \frac{b_i - a_i}{2\alpha} \; ; \; \frac{a_i + b_i}{2} + \frac{b_i - a_i}{2\alpha} \right]. \tag{1.2}$$

The same experiments as in Table 1.10 were conducted with these tighter time windows. Figure 1.12 represents the routing costs of the solutions of the VRPTW (red line) and the VRPDO (blue line), respectively, when the width of time windows at individual locations varies.

The x-axis represent the width of time windows and the y-axis shows the routing costs The time windows width was set by applying a factor α going from 1 and 10 in equation (1.2), leading to time window width between 20 and 180 minutes. Note that some instances become infeasible when the time windows are too tight. Thus, not all instances are taken into account in this figure. The main conclusion of this figure is that the cost of both the VRPTW and the VRPDO increase when time windows are tightened, but this increase is only 10% in the case of the VRPDO. Similar managerial insights were also reported in Buzzega and Novellani (2021)
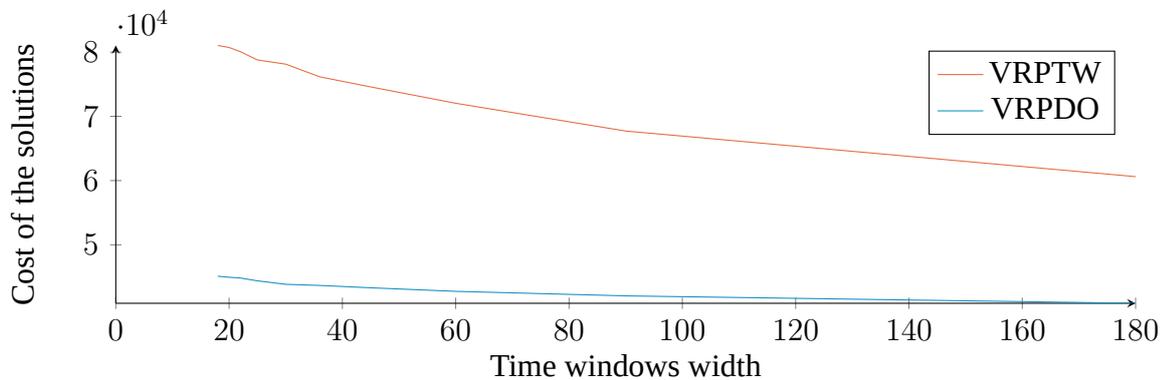
Figure 1.12 – Impact on time windows on the routing costs

on the single vehicle case.

Figure 1.13 represents the number of vehicles used in the solution of the VRPTW (red line) and the VRPDO (blue line), respectively, when the width of time windows at individual locations varies between 20 and 180 minutes.
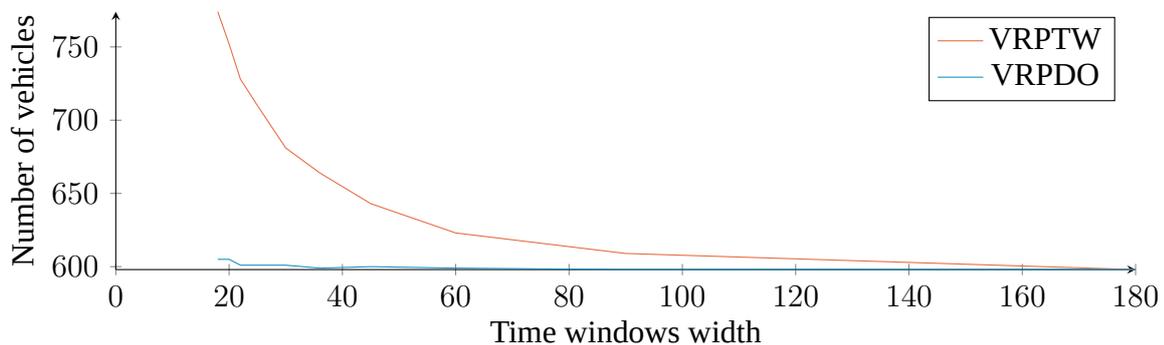
Figure 1.13 – Impact of time windows on the fleet size

With large time windows, each route can deliver many customers, so that the bottleneck is the capacity of truck. This is the solutions of both the VRPTW and the VRPDO use 598 vehicles. With very tight time window, the number of vehicles needed in the VRPTW grows dramatically, while it increases from 598 to 604 for the VRPDO. The main conclusion is that considering delivery options and SDLs makes it possible to serve customers at individual delivery locations with very narrow time windows, at the cost of minor increase of the vehicle fleet and very reasonable cost increase.

To illustrate this property, let us consider instance U_100_1, with 100 customers, small trucks, and many small lockers. When the widths of original time windows are divided by 10, the number of vehicles remains the same, with an average fill rate of 90%. The routing costs

increase by only 2.8%. A first explanation is that the number of customers served at an SDL is 65 and 66, respectively, and these locations are not impacted by the time window reduction. A second explanation is that 14 customers are delivered at another location and in another time window in both cases. Finally, note that the general service level constraints are binding in the two scenarios.

Delivery options and service level constraints

In order to quantify the impact of service level constraints, different values of parameters β_1 and β_2 were tested. Table 1.11 presents sum of the routing cost of all VRPDO instances, with respect to the required service level. Columns 3, 4 and 5 correspond to solutions i which the ratio of customers served with their preferred option (β_1) is at least 70, 80 and 90, respectively. Lines 3 to 6 correspond to the solutions where the ration of customers delivered with options 1 or 2 (β_2) is equal to at least 80, 90, 95 and 100, respectively.

		% of customers served with option 1 (β_1)		
		70%	80%	90%
% of customers served with options 1 or 2	80	60 215.29	61 475.93	NA
	90	60 393.85	61 907.68 *	66 155.67 *
	95	60 710.38	61 980.41 *	66 574.68 *
	100	61 892.21	63 172.80 *	-

Table 1.11 – Impact of service level on the routing costs

In most solutions of Table 1.11, the total number of vehicles used is 866. Sometimes, one additional vehicle is necessary ; in these cases, the routing costs are followed by a *. The cell corresponding to $(\beta_1, \beta_2) = (90, 80)$ is left empty because the corresponding values of β_1 and β_2 are incompatible. The cell of $(\beta_1, \beta_2) = (90, 100)$ is left empty because some instances are infeasible with such restrictive parameters.

This table shows that the service level can be greatly improved with no major increase of the routing costs. Moreover, the service level has very limited impact on the fleet size. The impact of β_2 seems to be rather limited. Indeed, the average cost increase when β_2 varies from 80 to 100 (with $\beta_1 = 70$) is only 2.8%, whereas the cost increase when β_1 varies from 70 to 90 (with $\beta_2 = 90$) is 9.5%.

1.8   Conclusion

In this chapter, we proposed a new extension of the VRPTW allowing multiple delivery options for each customer. The Vehicle Routing Problem with Delivery Options (VRPDO) includes the use of shared delivery locations, such as lockers, and takes into account customer preferences. In addition, the model is general enough to include new modes of delivery, such as trunk delivery. The VRPDO is theoretically challenging because it introduces synchronized resources and a new structure of the search space, due to the presence of several delivery locations for each customer.

The numerical experiments show that VRPDO can save considerable amounts of money compared with VRPTW. Moreover, for a small cost increase, very high quality of service can be achieved, especially with respect to the time window width.

To solve this problem we developed a LNS meta-heuristic based on small destructions: Small and Large Neighborhood Search (SLNS). After implementing a large number of ruin and recreate operators, a comprehensive tuning process resulted in the selection of a few relevant operators. The experiments show that combining local impact fast operators and global impact slower operators is an efficient strategy. A key success factor of the underlying SLNS is the tremendous number of iterations that can be performed thanks to the small destroy moves, while the search is diversified by larger destroy moves if necessary. A comparison with previously developed LNS metaheuristics on the VRPDO show that this combination of small and large destruction provides much better results, especially for large instances.

HYBRID LARGE NEIGHBORHOOD SEARCH HEURISTICS FOR GENERALIZED VEHICLE ROUTING PROBLEMS

2.1 Introduction

In this chapter, we present a matheuristic based on SLNS which is used to solve several vehicle routing problem with time windows and delivery options, generalizing the *generalized vehicle routing problem with time windows* (GVRPTW, Moccia et al., 2012).

The most general considered problem is the *vehicle routing problem with delivery options* (VRPDO). It extends the GVRPTW by service-level constraints defined by the customers' preferences for options and by location capacities that must be respected. It also generalizes other problems such as the *vehicle routing problem with roaming delivery locations* (VRPRDL, Reyes et al., 2017) in which options result from deliveries to a customer's car trunk, and the *vehicle routing problem with multiple time windows* (VRPMTW, Favaretto et al., 2007) in which all options of a customer refer to the same physical location but have disjoint time windows.

The contribution of this work is the design of two exact components and their integration in SLNS. The result is a powerfull matheuristic that offers good performances on all GVRPTW variants mentionned above. First, a *set-partitioning problem* (SPP) is solved periodically to combine previously found routes into new solutions. Second, the *Balas-Simonetti* (BS, Balas and Simonetti, 2001) neighborhood is adapted to further improve already good solutions. The BS neighborhood is an exponentially-sized neighborhood in which a best improving solution can be found by solving a shortest-path problem in a layered graph. The two components are optional, giving rise to four different *configurations* of the matheuristic.

The experimentation with the two exact components complementing SLNS follows the general trend towards hybridization and matheuristics for difficult VRP variants. For example, Tellez et al. (2018) solve a set-partitioning problem in an LNS for a time-consistent dial-a-ride problem, Toffolo et al. (2019) use the BS neighborhood in a structural decomposition approach for the *capacitated vehicle routing problem,* and Yuan et al. (2021) combine a route-based model with dynamic programming and local searches for the GVRPTW. Our idea behind the two exact components in SLNS is that delivery options require a 'global view' on the search space, because the usefulness of an option is observable only when it is combined with suitable other options. Thus, simple local modifications of a solution alone will typically not show the usefulness of an option. What is required is larger modifications that result from several simultaneous changes in the assignment of customer requests to routes, the selection of possible options, and the routing, i.e., sequence in which deliveries are performed.

In an exhaustive computational study, we evaluate the four configurations of our matheuristic on several benchmark instances of the GVRPTW, VRPRDL, VRPMTW, and VRPDO. On all

benchmark sets, at least one configuration of the matheuristic, often several, or even all configurations are competitive with the previous state-of-the-art methods. For the GVRPTW, the four configurations of our matheuristic work almost equally well: compared to the state of the art, they find equivalent or slightly improving results with less computational effort. For the VRPMTV, the comparision with the literature is even tighter: only two variants of our matheuristic outperform the ALNS competitor improving gaps only marginally. Compared to a recent exact branch-price-and-cut algorithm for the VRPDL and VRPHDL, our best matheuristic configuration finds all known optimal solutions except one and provides new best-known solutions for all instances that were not solved to proven optimality. Finally, significant improvements are provided for the VRPDO.

One might argue that a single matheuristic instead of four different matheuristic configurations would be preferable. However, as another contribution of our work, we will clearly work out characteristics of GVRPTW variants and matheuristic components that either fit well together or are cumbersome.

The remainder of this chapter is organized as follows. Section 2.2 formally defines the GVRPTW variants. We describe the two exact components, i.e., the SPP formulation and the BS neighborhood in sections 2.3 and 2.4 respectively. Section 2.5 presents the computational experiments and their results. Final conclusions are drawn in Section 2.6.

2.2 Generalized vehicle routing problems variants

In this section, we formally introduce the considered problem variants. We start with a reminder of the notations used for the VRPDO, introduced in Chapter 1, because it is the most general variant. Afterwards, GVRPTW, VRPRDL and VRPMTW are briefly described and it is explained how they can be modeled as special cases of the VRPDO. We also discuss of the previous paper considering these problems to which we will compare our method.

2.2.1 Vehicle Routing Problem with Delivery Options

The VRPDO is the problem of selecting delivery options, exactly one for each customer, and determining a cost-minimal set of feasible routes that serve the selected delivery options while respecting location-capacity and service-level constraints.

Let N be the set of customers, L be the set of locations, and $P = \{1, 2, \ldots, \bar{p}\}$ be the priority levels. A delivery option is a triple composed of a customer, location, and priority level. Formally, let $O \subset N \times L \times P$ be the set of delivery options. For an option $o \in O$, we write $n_o \in N$ for its customer/delivery request, $l_o \in L$ for its location, and $p_o \in P$ for its priority level. Additionally, each option o has a service time s_o.

A request $n \in N$ is characterized by a demand q_n. The request is served by choosing one of the options $O_n = \{(n_o, l_o, p_o) \in O : n_o = n\}$.

We define $O_l = \{(n_o, l_o, p_o) \in O : l_o = l\}$ as the set of options belonging to location $l \in L$. Let $L^m = \{l \in L : |O_l| > 1\}$ be the set of shared delivery locations. Shared locations $l \in L^m$ have a limited capacity C_l in terms of the number of shipments that can be delivered there. Moreover, we assume that at all locations $l \in L$ have an associated time window $[a_l, b_l]$ that describes the time period in which deliveries can be performed.

A fleet of K homogeneous vehicles with capacity Q is housed at the depot location $l_0 \in L$. For each pair of locations l and $l' \in L$, the travel time $t_{ll'}$ and the travel cost $c_{ll'}$ are given.

Service-level constraints are modeled with numbers $\beta_p \in [0, 1]$ for $p \in P$. The value β_p is the minimum percentage of options of service level not greater than p that must be chosen.

A route $r = (0, o_1, ..., o_h, 0)$ is as sequence of options in which the artificial options $o_0 = 0$ and $o_{h+1} = 0$ represent the visit of the depot location ℓ_0 at the start and end of the route, respectively. The demand served by route r is $q(r) = \sum_{j=1}^{h} q_{n_{o_j}}$, so that r is capacity-feasible if $q(r) \leq Q$ holds. A route is time-window feasible if there exists a schedule $(T_0, T_1, ..., T_h, T_{h+1}) \in \mathbb{R}^{h+2}$ which complies with the option service times, travel times, and time windows, i.e., if $T_{j-1} + t_{\ell_{o_{j-1}}, \ell_{o_j}} + s_{o_{j-1}} \leq T_j$ for all $1 \leq j \leq h+1$ (assuming $s_{o_0} = 0$)

and $[T_j, T_j + s_{o_j}] \subseteq [a_{\ell_{o_j}}, b_{\ell_{o_j}}]$ for all $0 \leq j \leq h + 1$. A route r is feasible if it fulfills both capacity and time-window constraints. The cost of a route is the sum of the travel cost between the consecutively visited locations, i.e., $c_r = \sum_{j=1}^{h+1} c_{\ell_{o_{j-1}}, \ell_{o_j}}$.

A solution S to VRPDO is a set of feasible routes that selects exactly one option per customer. Let $O(S)$ be the set of options selected in the solution S. Then, the requirement that S selects exactly one option per request translates into $|O(S) \cap O_n| = 1$ for all $n \in N$. The solution fulfills the location capacity constraints if $|O(S) \cap O_l| \leq C_l$ for all $l \in L^m$. It fulfills the service-level constraints if, for all $p \in P$:

$$|\{o \in O(S) : p_o \leq p\}| \geq \beta_p |N| \qquad \Leftrightarrow \qquad |\{o \in O(S) : p_o > p\}| \leq (1 - \beta_p)|N| \quad (2.1)$$

2.2.2 Generalized Vehicle Routing Problem with Time Windows

The GVRPTW (Moccia et al., 2012) is a direct generalization of the vehicle routing problem with time windows (VRPTW, Desaulniers et al., 2014; Savelsbergh, 1985), in which customers have to be served at one of their potential delivery locations respecting the corresponding time window. Each potential delivery location defines a delivery option for the customer. Thus, the GVRPTW can be modeled and solved as a VRPDO without synchronized resources.

Moccia et al. (2012) first designed a heuristic solution of the GVRPTW. They propose an incremental tabu search using a dynamic-programming component that allows changing customers' locations when inserting a customer in a route. The tabu search provides solutions for instances with up to 120 customers in a few hundred seconds. In addition, these instances were recently solved by Yuan et al. (2021) with a column-generation based heuristic. On average, the cost of the solutions was reduced by 0.17%, by 10 new best solutions out of the 20 instances, with a computing time nearly divided by 2 (by taking into account the difference of processors). The method presented by Ozbaygin et al. (2017) could deal with these instances, but they do not present results.

2.2.3 Vehicle Routing Problem with (Home and) Roaming Delivery Locations

The VRPRDL, introduced by Reyes et al. (2017), specifically models the delivery to the trunk of cars. Customers must specify where they plan to be during the planning horizon, thereby defining different delivery options. The VRPRDL can be seen as a special case of the GVRPTW

with non-overlapping time windows for the delivery options of each customer (Ozbaygin et al., 2017).

The VRPHRDL is an extension of the VRPRDL with an additional so-called *home option* for each customer (Ozbaygin et al., 2017). This additional delivery option has a non-constraining time window, e.g., identical to the planning horizon. Both problems are special cases of the GVRPTW and can, therefore, be modeled and solved as VRPDO without synchronized resources.

Reyes et al. (2017) propose a variable neighborhood search to solve the VRPRDL. It embeds a dynamic programming algorithm to optimize the travel distance of a given customer sequence. Ozbaygin et al. (2017) develop a branch-and-price algorithm to solve the VRPRDL and VRPHRDL with up to 120 customers. Yuan (2019) (Yuan et al., 2021) propose a heuristic-based branch-and-price that solves instances with up to 120 customers, and find better results than Ozbaygin et al. (2017) for 19 instances. In addition, Lombard et al. (2018) solve smaller instances of a stochastic VRPRDL.

2.2.4 Vehicle Routing Problem with Multiple Time Windows

The VRPMTW was introduced by Favaretto et al. (2007) as an extension of the VRPTW where each customer can have multiple time windows. If one considers each time window as a different option, the VRPMTW is a special case of the VRPDO. All delivery locations of a customer are at the same physical place. Options of two customers, however, always refer to different physical places.

The objective of the VRPMTW is either minimizing the travel distance or travel duration (total travel, service, and waiting time). In both cases, a fixed cost per route is included in the objective function.

Belhaiza et al. (2017) propose a hybrid variable neighborhood tabu search and a set of benchmark instances with 100 customers. A revised version of the approach was described in Belhaiza et al. (2017) as a hybrid genetic variable neighborhood search. Larsen and Pacino (2019) solve the VRPMTW with an adaptive LNS with a problem-tailored insertion procedure. They generalize the forward time slack procedure of Savelsbergh (1992) to take into account all the time windows of the visited customers. Thus, the time windows used to visit the customers of a route can be changed to insert a new customer in this route. Hoogeboom et al. (2020) describe an adaptive variable neighborhood search relying on a generalization of the forward time slacks for the VRPMTW with duration-minimization objective.

2.3 Set Partioning Problem

The utilization of the *Set Partitioning Problem (SPP)* to solve vehicle routing problems was first introduced by Foster (1976). It is nowadays widely used in column-generation approaches for many variants of the VRP (Toth and Vigo, 2002). It can also be used to recombine routes that are produced by a heuristic. The SPP is used as a post-optimization technique (Gschwind and Drexl, 2019; Mancini, 2017; Rochat and Taillard, 1995), as well as inside hybrid heuristics (Mendoza and Villegas, 2013; Parragh and Schmid, 2013; Prescott-Gagnon et al., 2009; Subramanian et al., 2013a; Tellez et al., 2018).

Let R be a set of feasible routes. For each route $r \in R$ and each option $o \in O$, the binary parameter α_r^o takes value 1 if the option $o \in O$ is served by the route $r \in R$, and 0 otherwise. The model uses binary variables λ_r for $r \in R$ that indicate if route r is used in the solution:

Model 2: route-based model

$$\min \sum_{r \in R} c_r \lambda_r \tag{2.a}$$

$$s.t. \sum_{r \in R} \sum_{o \in O_n} \alpha_r^o \lambda_r = 1 \qquad \forall n \in N \tag{2.b}$$

$$\sum_{r \in R} \sum_{o \in O_l} \alpha_r^o \lambda_r \leqslant C^l \qquad \forall l \in L^m \tag{2.c}$$

$$\sum_{r \in R} \sum_{o \in O : p_o > p} \alpha_r^o \lambda_r \leqslant (1 - \beta_p)|N| \qquad \forall p \in P \setminus \{\bar{p}\} \tag{2.d}$$

$$\sum_{r \in R} \lambda_r \leqslant K \tag{2.e}$$

$$\lambda_r \in \{0, 1\} \qquad \forall r \in R$$

The objective (2.a) minimizes the total cost of the solution, i.e., the sum of the cost of the routes used. The set-partitioning constraints (2.b) state that each customer must be served exactly once. The capacity constraints for shared locations are given by (2.c) and the service-level constraints by (2.d). The fleet-size constraint (2.e) sets the upper bound K on the number of routes used in the solution. Finally, the variable domains are given.

For solving the model with a MIP solver, it is known that typically set-covering formulations have shorter computation times than the respective set-partitioning formulations (Yıldırım and Çatay, 2015). If the travel times and costs fulfill the triangle inequality (always true for the benchmarks of Section 2.5), the partitioning constraints can be replaced by covering constraints, i.e., ≥ 1 instead of $= 1$ in (2.b). In a set-covering solution, more than one option may be used

to serve a customer. A simple greedy procedure can quickly repair and improve solutions which overcover some customers. Note that the repaired solutions automatically respect the location-capacity and service-level constraints.

To further reduce the solution times, we extend the formulation by introducing binary variables y_o that indicate if option $o \in O$ is selected. We prioritize branching on these variables in the MIP solver. The new y variables are coupled with the route variables via the following constraint

$$\sum_{r \in R} \alpha_r^o \lambda_r = y_o \qquad \qquad \forall o \in O. \qquad (2.f)$$

To integrate the SPP solving into SLNS, we extend the procedure of Tellez et al. (2018) used in their LNS-SPP for the fleet size and mix dial-a-ride problem with reconfigurable vehicle capacity: Initially, the SPP is solved every 1000 iterations. If the solver fails to prove optimality twice in a row the time between two calls is reduced by a quarter.

In the scope of SLNS, R is the pool of routes. This data-structure stores the routes of the solutions generated by SLNS, whether or not they are accepted. We use the size $|R|$ to decide on when to solve Model 2 with the MIP solver, and the pool R is emptied after each call to the SPP component. More precisely, it is solved when R contains at least $\max(100, 38000 - 180|N|)$ different routes. This threshold size is increased by 60% if the solver has proven optimality twice in a row. Conversely, it is reduced by the same factor if the solver has failed to prove optimality twice in a row. In addition, it is solved only if the cost of the best-known solution has improved by less than 1% over the last 5ω iterations (i.e. five times the number of iterations without improvement before performing a large destruction, see Section 1.5). Besides, during the experiments to tune these parameters, we noticed that it is more efficient to decide on when to use the SPP based on the size of R rather than based on the number of iteration during which R was filled.

2.4 Balas-Simonetti neighborhood

Balas (1999) proposed and analyzed a family of large-scale neighborhoods that, for the *asymmetric traveling salesman problem* (ATSP), can be searched efficiently (in linear time in the size of the Hamiltonian path). The neighborhoods $\mathcal{N}_{BS}^k$ are parameterized by an integer $k \geq 2$ and Balas and Simonetti (2001) used them within a local-search algorithm for the ATSP and the ATSP with time windows.

We very briefly summarize the *Balas-Simonetti neighborhood* for the ATSP: Given an TSP Hamiltonian path $x = (x_0, x_1, \ldots, x_n, x_{n+1})$ the neighborhood $\mathcal{N}_{BS}^k(x)$, for a given value $k \geq 2$, consists of all tours $x' = (x_0, x_{\pi(1)}, \ldots, x_{\pi(n)}, x_{n+1})$, where π is a permutation of $\{1, \ldots, n\}$ that fulfills the following conditions: For any two indices $i, j \in \{1, \ldots, n\}$ with $i + k \leq j$, the inequality $\pi(i) < \pi(j)$ holds. It means that if a vertex x_i precedes a vertex x_j by at least k positions in the given path x, then x_i must also precede x_j in the neighbor x'. For a given value of the parameter k, a best neighbor $x' \in \mathcal{N}_{BS}^k(x)$ can be determined in $O(k^2 2^{k-2} n)$ by solving a shortest-path problem in an auxiliary network (Balas, 1999), i.e., for a fixed value of the parameter k the neighborhood exploration is linear in n.

An example of an auxiliary network is depicted in Figure 2.1 for $k = 3$ and $n = 5$. The graph consist of layers (depicted from left to right in the figure) that correspond to the positions in the Hamiltonian path. The vertices at each layer can be ordered into rows that are associated with an offset value α. In the following, the vertices of the auxiliary network are called *states*. Thus, each state in the auxiliary network is associated with the vertex $x_{j+\alpha}$ where j is the layer of the state and α the value associated with the state. Arcs exclusively go from states of a layer j to states of the subsequent layer $j + 1$. The general structure of the auxiliary network is described in several works, e.g., (Balas and Simonetti, 2001) and several subsequent articles (Hintsch and Irnich, 2018; Irnich, 2008b; Tilk and Irnich, 2017).

Every source-sink path, from 0 to $n+1$, in the auxiliary network corresponds to a neighbor x'. For example, the green sequence of states at the top row in Figure 2.1 corresponds to $x' = x = (0, 1, 2, 3, 4, 5, 6)$. The red sequence of states corresponds to the neighbor $x' = (0, 2, 3, 1, 5, 4, 6)$, i.e., a Hamiltonian path that respects the precedence constraint with respect to the initial tour x and the given parameter $k = 3$.

The structure of the auxiliary network, i.e., states and connecting arcs, depends only on k and n, but not on the given path x. The only difference between two auxiliary networks is the cost of the arcs that must be set to the distance between the considered customers. Consequently, if the BS neighborhood must be explored multiple times, the auxiliary network can be kept. Moreover,
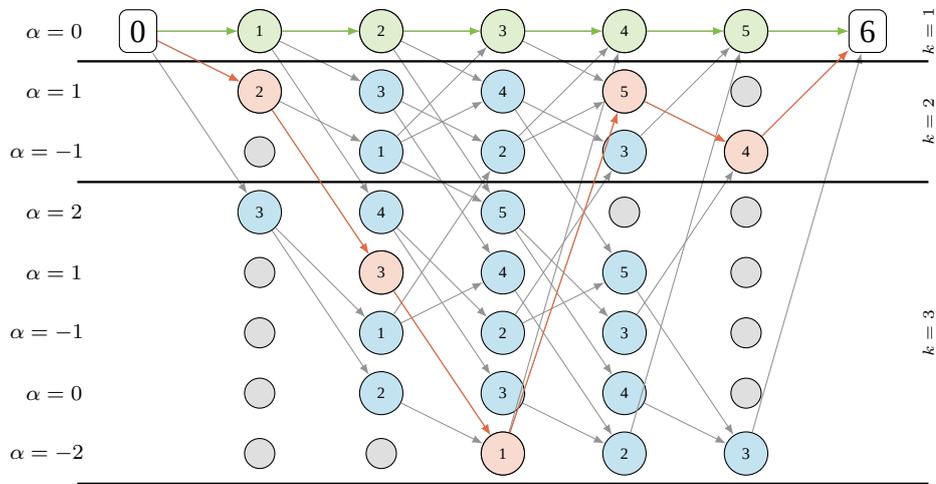
Figure 2.1 – Example of the auxiliary network for $k = 3$ and $n = 5$

the auxiliary network for k' is always a state-induced subgraph of the auxiliary network for any larger $k > k'$ (indicated by the separating lines in Figure 2.1).

Adaptions. Next, we explain how to adapt the Balas-Simonetti neighborhood such that it can be used for all GVRPTW variants, including the VRPDO with the inter-route synchronization constraints imposed by location capacities and service-level requirements.

First, we generalize the auxiliary network such that it can deal with options. Contrary to the TSP, in the VRPDO, a delivery request n can be served using the different delivery options O_n. Hence, *meta states* encompass sets of states that represent all options of the respective customer, as shown in Figure 2.2. All states for options of a customer are linked to all states for options of the subsequent customer.

Figure 2.2 – A sequence of customers in the auxiliary network for the ATSP and its generalization to delivery options in the GVRPTW

Second, a *shortest-path problem with resources constraints* (SPPRC, Irnich and Desaulniers, 2005) must be solved on the generalized auxiliary network, in contrast to solving one without resource constraints for the ATSP. We solve the corresponding SPPRCs with a label-setting

algorithm as follows: A label $\mathcal{L}_i = (i, C_i, T_i, Q_i, (R_i^p)_{p \in P}, (H_i^l)_{l \in L})$ represents a partial path from the depot 0 to a state i in the auxiliary network. Recall that a state i represents an option and therefore a delivery request n_i, a location l_i, a priority level p_i, and a service time s_i are associated. The components of the label $\mathcal{L}_i$ are

  (i) last state i of the partial path,

 (ii) the accumulated routing cost C_i,

(iii) the earliest arrival time T_i at the location associated with i,

 (iv) the accumulated demand (=load) Q_i in the vehicle,

  (v) the number H_i^l of shipments for each shared delivery location (for all $l \in L^m$), and

 (vi) the number R_i^p of options o with priority p_o greater than p served along the partial path (for all $p \in P \setminus \{\bar{p}\}$).

The initial partial path in the auxiliary network starts from the depot 0 and the label $\mathcal{L}_0 = (0, 0, a_{l_0}, 0, \mathbf{0}, \mathbf{0})$. The label-setting algorithm extends labels along the arcs of the auxiliary network. Extending a label $\mathcal{L}_i = (i, C_i, T_i, Q_i, (R^p)_i^{p \in P \setminus \{\bar{p}\}}, (H_l)_i^{l \in L^m})$ along an arc (i, j) results in the label $\mathcal{L}_j = (j, C_j, T_j, Q_j, (R^p)_j^{p \in P \setminus \{\bar{p}\}}, (H_l)_j^{l \in L^m})$ defined by:

$$C_j = C_i + c_{l_i l_j}$$
$$Q_j = Q_i + q_{n_j}$$
$$T_j = \max\{a_{l_j}, T_i + t_{l_i l_j} + s_i\}$$
$$H_j^l = \begin{cases} H_i^l + 1, & \text{if } l_j = l \\ H_i^l, & \text{otherwise} \end{cases} \qquad \text{for all } l \in L$$
$$R_j^p = \begin{cases} R_i^p + 1, & \text{if } p_j > p \\ R_i^p, & \text{otherwise} \end{cases} \qquad \text{for all } p \in P \setminus \{\bar{p}\}$$

The extension is feasible, if the following constraints

$$Q_j \leq Q$$
$$T_j \leq b_j$$
$$H_j^l \leq C_l - \bar{H}^l, \qquad\qquad\qquad \text{for all } l \in L^m$$
$$R_j^p \leq (1 - \beta_p)|N| - \bar{R}^p, \qquad\qquad \text{for all } p \in P \setminus \{\bar{p}\}$$

are fulfilled, where the values $\bar{H}^l$ for $l \in L$ and $\bar{R}^p$ for $p \in P \setminus \{\bar{p}\}$ are parameters that control

the required slack in the location-capacity and service-level constraints.

The presented adaptation let us use the Balas-Simonetti neighborhood on the *TSPDO*, but not on the VRPDO. We consider three different use cases for the BS component with multiple routes:

(1) the local optimization of a single route

(2) the local optimization of a giant route that aggregates all routes of the solution

(3) the simultaneous local optimization each pairs of routes

In case of a single route $r \in s$ taken from a current solution s, the parameters $\bar{H}^l$ and $\bar{R}^p$ for $p \in P \setminus \{\bar{p}\}$ can be set to the resource consumptions of all other routes $r' \in s \setminus \{r\}$.

In the *giant route* case, all routes of a given solution s need to be joined as in (Irnich, 2008a) and (Hintsch and Irnich, 2018). The parameters $\bar{H}^l$ and $\bar{R}^p$ are set to 0. On arcs $(0,0)$ joining two different routes, the resources T and Q are reset to a_{l_0} and 0, respectively. In contrast, the resources H and R are kept at their current value enabling that the inter-route synchronization constraints are incorporated correctly.

In case of pairs (r_1, r_2) of routes, the two routes are joined leading to a partial giant route. The parameters $\bar{H}^l$ and $\bar{R}^p$ are then set to the resource consumptions of all other routes $r' \in s \setminus \{r_1, r_2\}$. Note that the BS neighborhood here also allows moving the depot vertices (in the middle) so that the route length of r_1 and r_2 can change.

Since all resource extensions are non-decreasing and resource consumptions are bounded from above, we can apply standard $\leq$-dominance (Irnich, 2008b). Obviously, resource H and R are obsolete in the GVRPTW, VRPRDL, VRPHRDL, and VRPMTW. However, due to a possibly large number of capacitated shared delivery locations and preference levels in the VRPDO, the dominance relation can become rather weak in this variant.

We have conducted some preliminary experiments that have shown that the route-pairing version is the most effective (note that it includes the single route optimization). Solving the SPPRC for the giant route is excessively time-consuming, even on small instances and for small BS parameters $k \geq 2$. We therefore use the BS component only for all pairs of routes in the following.

Acceleration Techniques.   As computing time is essential, we devise six different acceleration techniques, two exact (they do not change the output of the labeling algorithm) and four heuristic acceleration methods (they may hinder the labeling algorithm to compute an optimal solution of the SPPRC).

On the exact side, we first use a bidirectional labeling algorithm (Righini and Salani, 2006). Note that backward extension and merging of labels follows standard rules (Irnich, 2008a).

Second, when pairing routes, we solve an SPPRC on every pair of routes in the solution, i.e., each route will be used in several combinations. Therefore, some of the labels computed at the beginning of the forward and backward labeling are identical for pairs that share a common route and must be computed only once.

On the heuristic side, we first restrict the labeling to a limited discrepancy search (Feillet et al., 2007). Therein, we limit the number of customers that can change their delivery option. The consequence is that all customer requests can still be permuted but only a limited number of the currently used options can be replaced by an alternative option of the respective request. In our experiments, we limit the number of customers that can change their delivery option to 3.

Second, we use a heuristic bounding strategy that discards labels if their cost is larger than some threshold value. The threshold value is computed as the difference between the best known cost and the sum of the best individual service cost of each non-served customer. For each solution computed in the course of the new LNS, we compute a *customer-specific* service cost of each customer and update the best one if necessary: If a customer is served in an individual delivery location, its customer-specific service cost is the mean of the costs of the ingoing and outgoing arcs used to access it. If a customer is served in a shared delivery location, the cost of the ingoing and outgoing arcs is divided by two times the number of customers served at this location by this route.

Third, a significant number of the resources of a label is dedicated to the capacity consumption of shared delivery locations, which results in a relatively weak dominance. To strengthen the dominance, we ignore some of the shared delivery locations in the dominance test. To decide which shared delivery locations are ignored, the cost of the best solution that completely utilizes the capacity of a location is recorded in the LNS. We ignore those locations used only in solutions with a cost that is more than 10 % higher than the cost of the current best-found solution.

Fourth, routes are sequences of options and multiple options taking place at the same shared delivery location can be swapped without changing the cost of the solution. We break these symmetries by maintaining the order of the currently chosen options of the same shared delivery locations.

Both exact components are only applied if the cost of the best-known solution has improved by less than 1% over the last 5ω iterations. Moreover, a good solution-quality-to-time compro-

mise is a small value of $k = 5$. With this value the BS component is applied exclusively to solutions that serve all customers (i.e with an empty request bank B_s) and with a cost smaller than 1.01 times of the cost of the best-known solution. These parameters were obtained from preliminary experiments.

2.5 Computational Experiments

To summarize, let us present Algorithm 3 which is the outline of SLNS with the two exact components. It differs from Algorithm 2 by the following instructions:

Line (16) applies the Balas-Simonetti neighborhood on the current solution if it is promising.

Line (19) saves the routes of the current solution in the pool R.

Line (28) applies the SPP to recombine the routes of R and improve solution s.

Algorithm 3: Small and Large Neighborhood Search with Set Partitioning Problem and Balas-Simonetti neighborhood

---
```
 1: s : initial solution
 2: iter = 0, R ← ∅
 3: while the time budget is not reached do
 4:     randomly select a recreate operator σ⁺ ∈ Σ⁺
 5:     if iter < ω then
 6:         s′ ← s
 7:         randomly select a local ruin operator σ⁻ ∈ Σ⁻|local
 8:         randomly select a destruction size Φ ∈ [δsmall, Δsmall]
 9:     else
10:         iter = 0
11:         s′ ← s*
12:         randomly select a ruin operator σ⁻ ∈ Σ⁻
13:         randomly select a destruction size Φ ∈ [δlarge, Δlarge]
14:     end if
15:     s′ ← σ⁺(σ⁻(s′, Φ))
16:     if f(s′) < (1 + ε)f(s*)  and  s′ is feasible then
17:         improve s′ with the Balas-Simonetti neighborhood
18:     end if
19:     add routes of s′ to pool R
20:     iter ← iter + 1
21:     if f′(s′) < f′(s) or iter = ω then
22:         s ← s′
23:     end if
24:     if f(s′) < f(s*) and Bs′ = ∅ then
25:         s* ← s′
26:         iter ← 0
27:     end if
28:     if a sufficient number of routes is generated then
29:         improve s with the SPP and route set R
30:         R ← ∅
31:     end if
32: end while
33: return s*
```

Here line numbers correspond to: $\sigma^+ \in \Sigma^+$ (line 4), $iter < \omega$ (line 5), $\sigma^- \in \Sigma^-|_{\text{local}}$ (line 7), $\Phi \in [\delta_{\text{small}}, \Delta_{\text{small}}]$ (line 8), $\sigma^- \in \Sigma^-$ (line 12), $\Phi \in [\delta_{\text{large}}, \Delta_{\text{large}}]$ (line 13), $s' \leftarrow \sigma^+(\sigma^-(s', \Phi))$ (line 15), $f(s') < (1+\epsilon)f(s^*)$ (line 16), $f'(s') < f'(s)$ (line 21), $f(s') < f(s^*)$ and $B_{s'} = \emptyset$ (line 24).

In the following, we evaluate four configurations of Algorithm 3:

SLNS without any exact component $\hspace{4cm}$ Lines (16), (19), and (28) deactivated

SLNS+BS with only the BS component $\hspace{4cm}$ Lines (19), and (28) deactivated

SLNS+SPP with only the SPP component $\hspace{5cm}$ Line (16) deactivated

SLNS+SPP+BS with both SPP and BS

In this section, we report the results of computational experiments that were conducted on GVRPTW, VRPRDL, VRPHRDL, VRPMTW, and VRPDO benchmarks with the four configurations of the matheuristic. The algorithms have been coded in C++ and compiled into 64-bit single-thread code with g++ 5.4.0. All experiments were performed using Linux, Ubuntu 16.04 LTS, running on an Intel Xeon X5650 @ 2.57 GHz. We use IBM Ilog CPLEX 12.8.0 (IBM, 2018) to solve the SPP and activate the options `branch_up_first` and `emphasis_on_hidden_feasible_solutions`.

A comparison with algorithms from the literature on standard benchmark sets for the GVRPTW, VRPRDL, VRPHRDL, VRPMTW, and VRPDO is conducted in Sections 2.5.1 to 2.5.4, respectively. A synthesis of numerical results is proposed in Section 2.5.5. Finally, Section 2.5.6 presents additional experimental results. Our results on the VRP(H)RDL and the GVRPTW are compared to those recently published in Yuan et al. (2021), furthermore, the scaling capabilities of our method are evaluated on 400 customers VRPDO instances.

The tables presented in the following sections provide the following information:

Instance(s): Name of the benchmark instance (group)

#: Number of instances in the group

$|N|$: Number of customer requests

#veh: Upper bound computed/set on the number of vehicle/the fleet size

Σveh: Sum of the number of vehicles over the group of instances

Cost: Overall cost of the best obtained solution

ΣCost: same, summed over the group of instances

#Best: Number of instances for which an algorithm has found a best-known solution

GAP: gap between the best solution value found by our algorithm (UB) and the one from the literature (Sol) – computed as $100 \cdot (UB - Sol)/Sol$

$\emptyset$Gap: Average GAP over the respective instance group

2.5.1 Results for the GVRPTW

We compare all four configurations of SLNS with the *Iterative Tabu Search* (ITS) of Moccia et al. (2012) on their benchmark containing instances with up to 120 customers. These instances are named i-n-v_{min}-v_{max} where n is the number of customers, v_{min} the minimum, and v_{max} the maximum number of options per customer.

Moccia et al. (2012) consider the single objective of routing-cost minimization. For a fair comparison, we bound the number of available vehicles to the value reported in their work.

Moccia et al. (2012) performed their computational experiments on an Intel Core Duo @ 1.83 GHz and limited their algorithm to 10^5 iterations taking a total computation time of 8,105 seconds. In our matheuristic, we allocate a quota for the computation time for each instance according to the number of customers. In total, the SLNS consume 524 seconds for all 20 benchmark instances. According to PassMark-Software (2020), on single thread benchmarks, their processor is 2.18 times slower than ours. It means that, on average, the SLNS is configured to run approximately 7 times faster than the ITS.

	ITS		SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS	
Instance	#veh	Cost	Cost	Gap	Cost	Gap	Cost	Gap	Cost	Gap
i-030-04-08	4	3 498	**3 497**	-0.03	**3 497**	-0.03	**3 497**	-0.03	**3 497**	-0.03
i-030-08-12	4	2 866	**2 796**	-2.44	**2 796**	-2.44	**2 796**	-2.44	**2 796**	-2.44
i-040-04-08	6	**3 811**	**3 811**	0.00	**3 811**	0.00	**3 811**	0.00	**3 811**	0.00
i-040-08-12	6	**3 759**	3 768	0.24	3 768	0.24	3 768	0.24	3 768	0.24
i-050-04-08	8	5 447	5 447	0.00	5 447	0.00	**5 439**	-0.15	5 447	0.00
i-050-08-12	8	**4 034**	**4 034**	0.00	**4 034**	0.00	**4 034**	0.00	**4 034**	0.00
i-060-04-08	8	5 919	**5 908**	-0.19	5 926	0.12	5 919	0.00	5 926	0.12
i-060-08-12	8	**4 303**	**4 303**	0.00	**4 303**	0.00	**4 303**	0.00	4 332	0.67
i-070-04-08	10	**6 205**	6 246	0.66	6 246	0.66	6 224	0.31	6 246	0.66
i-070-08-12	10	4 645	**4 644**	-0.02	**4 644**	-0.02	**4 644**	-0.02	**4 644**	-0.02
i-080-04-08	12	7 425	**7 390**	-0.47	7 394	-0.42	7 394	-0.42	7 394	-0.42
i-080-08-12	12	5 734	5 686	-0.84	**5 661**	-1.27	5 692	-0.73	5 661	-1.27
i-090-04-08	11	**7 110**	7 187	1.08	7 182	1.01	7 187	1.08	7 215	1.48
i-090-08-12	11	5 810	5 903	1.60	5 869	1.02	5 830	0.34	**5 808**	-0.03
i-100-04-08	14	7 455	7 308	-1.97	7 308	-1.97	7 295	-2.15	**7 295**	-2.15
i-100-08-12	14	6 703	**6 546**	-2.34	**6 546**	-2.34	6 585	-1.76	6 606	-1.45
i-110-04-08	16	8 719	8 696	-0.26	**8 687**	-0.37	8 711	-0.09	**8 687**	-0.37
i-110-08-12	16	6 281	**6 249**	-0.51	6 487	3.28	6 338	0.91	6 310	0.46
i-120-04-08	15	8 512	**8 344**	-1.97	8 357	-1.82	**8 344**	-1.97	**8 344**	-1.97
i-120-08-12	15	6 833	6 829	-0.06	6 829	-0.06	**6 774**	-0.86	**6 774**	-0.86
Total	208	115 069	114 592	-0.38	114 792	-0.22	114 585	-0.39	114 595	-0.37
# Best		6	**11**		9		10		**11**	

Table 2.1 – Results for the GVRPTW benchmark of Moccia et al. (2012)

The results are presented in Table 2.1. Note that Moccia et al. (2012) does not indicate whether their results are the best results out of several runs. For the four SLNS configurations, the best solution out of five runs is taken into account. The best solution values are marked in bold.

All four SLNS configurations outperform the results of the ITS. The most best known solution (BKS) are found with SLNS and SLNS+SPP+BS. Regarding the routing cost, SLNS+SPP performs best and improves the results of ITS by 0.39% on average with a maximum improvement of 2.44%. Moreover, SLNS+SPP provides an equivalent or better solution than ITS on 15 of the 20 instances. However, all four configurations of SLNS perform rather similarly on the GVRPTW benchmark. In total, we provide 14 new best-known solutions. Detailed instance-by-instance results can be found in Appendix A.3.

2.5.2   Results for the VRPRDL and VRPHRDL

We next compare the four SLNS configurations with the exact branch-price-and-cut (BPC) algorithm of Tilk et al. (2020) on a benchmark set originally proposed by (Reyes et al., 2017). As suggested by Ozbaygin et al. (2017), the instances should be modified such that the triangle inequality for travel cost and travel times holds. The modified benchmark that we also use consists of 120 randomly generated instances with a size ranging from 15 to 120 delivery requests, with a maximum of 6 options per request. The benchmark set is divided into 60 VRPRDL and 60 VRPHRDL instances.

Note that we cannot fairly compare with the variable neighborhood search of Reyes et al. (2017), because we only have access to the modified instances provided by Ozbaygin et al. (2017). Since the BPC of Tilk et al. (2020) minimizes routing costs, we only focus on routing-cost minimization. To this end, we bound the number of vehicles by the number of routes given in their solutions.

The time budget for the four configurations of the SLNS is now set to 60 seconds for instances with up to 60 customers, and 300 seconds for the 120-customer instances. This is again a rather small computation time compared to the 2 hour and 6 hour time limit used in Tilk et al. (2020).

Aggregated results for the VRPRDL and the VRPHRDL are shown in Tables 2.2 and 2.3, respectively. Each line refers to a group of instances with identical number $|N|$ of customers. The BPC algorithm solves all 60 VRPRDL instances and 53 of 60 VRPHRDL instances benchmark set to proven optimality. The entry marked with [‡] indicates that best-known solution values have been used here, because optimality could not be proven by the BPC algorithm for seven instances

in this group. Again, the best solution values are marked in bold.

Comparing the four matheuristic configurations, SLNS+SPP performs best on both problem variants. For the VRPRDL, all optimal solutions could be found, while for the three other configurations the results are between 0.01 % and 0.04 % worse than the optimal solutions. Regarding the VRPHRDL, the four configurations improve the cost over the non-optimal solutions provided in Tilk et al. (2020). SLNS+SPP contributes with the largest improvement 0.36 % on average. Moreover, SLNS+SPP finds all but two known optimal solution values and the gap is 2.2 % for the instance group with 120 customers that contains all instances with unknown optimal solutions. Moreover, seven new best-known solutions have been obtained. Detailed results can be found in Appendix A.4.

			BCP		SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS	
Instances	#	$|N|$	Σveh	Σcost	Σcost	$\emptyset$Gap	Σcost	$\emptyset$Gap	Σcost	$\emptyset$Gap	Σcost	$\emptyset$Gap
1-5	5	15	24	**6 072**	**6 072**	0.00	**6 072**	0.00	**6 072**	0.00	**6 072**	0.00
6-10	5	20	27	**6 848**	**6 848**	0.00	**6 848**	0.00	**6 848**	0.00	**6 848**	0.00
11-20	10	30	68	**18 595**	**18 595**	0.00	**18 595**	0.00	**18 595**	0.00	**18 595**	0.00
21-30	10	60	129	**37 213**	**37 213**	0.00	**37 213**	0.00	**37 213**	0.00	**37 213**	0.00
31-40	10	120	189	**53 738**	53 759	0.04	53 876	0.22	**53 738**	0.00	53 761	0.04
41-50_v1	10	40	94	**29 838**	**29 838**	0.00	29 842	0.02	**29 838**	0.00	**29 838**	0.00
41-50_v2	10	40	74	**21 863**	**21 863**	0.00	**21 863**	0.00	**21 863**	0.00	**21 863**	0.00
Total	60		605	**174 167**	174 188	0.01	174 309	0.04	**174 167**	0.00	174 190	0.01
# Best				**60**	59		55		**60**		59	

Table 2.2 – Results for the VRPRDL benchmark of Ozbaygin et al. (2017)

			BCP		SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS	
Instances	#	$|N|$	Σveh	ΣCost	ΣCost	$\emptyset$Gap	ΣCost	$\emptyset$Gap	ΣCost	$\emptyset$Gap	ΣCost	$\emptyset$Gap
1–5	5	15	19	**5 450**	**5 450**	0.00	**5 450**	0.00	**5 450**	0.00	**5 450**	0.00
6–10	5	20	20	**5 604**	**5 604**	0.00	**5 604**	0.00	**5 604**	0.00	**5 604**	0.00
11–20	10	30	52	**15 128**	**15 128**	0.00	**15 128**	0.00	**15 128**	0.00	**15 128**	0.00
21–30	10	60	83	**26 800**	**26 800**	0.00	**26 800**	0.00	**26 800**	0.00	**26 800**	0.00
31–40	10	120	128	38 107[‡]	37 263	-2.12	37 349	-1.89	37 234	-2.20	**37 232**	-2.22
41–50_v1	10	40	87	**27 996**	**27 996**	0.00	**27 996**	0.00	**27 996**	0.00	**27 996**	0.00
41–50_v2	10	40	67	**20 958**	**20 958**	0.00	20 962	0.02	20 962	0.02	21 029	0.36
Total	60		456	140 043	139 199	-0.35	139 289	-0.31	**139 174**	**-0.36**	139 239	-0.31
					53		57		54		**58**		56

Table 2.3 – Results for the VRPHRDL benchmark of Ozbaygin et al. (2017)

2.5.3   Results for the VRPMTW

We compare the SLNS configurations with the ALNS of Larsen and Pacino (2019) on the adapted Solomon benchmark instances for the VRPMTW provided by Belhaiza et al. (2014). The algorithm of Belhaiza et al. minimizes the sum of the total routing cost and cost of the vehicles used. Since vehicle costs are fairly large (between 200 and 1000), we run Algorithm 2 twice, first to minimize the number of vehicles (which is then bounded), and second to minimize the routing cost, similarly to Ropke and Pisinger (2006a). The first phase tries to decrease the number of vehicles by removing the smallest route from the solution each time a feasible solution is found.

We report the best solution found out of 10 runs with an overall time budget of 600 seconds for both phases per instance. Larsen and Pacino (2019) used the same time limit and number of runs. Moreover, they performed their computational experiments on an Intel Core i7-4790K @ 4.00GHz, which is 2.02 times faster than our processor (see PassMark-Software, 2020).

				SLNS configurations							
		ALNS		SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS	
Group	#	Σveh	ΣCost	Σveh	ΣCost ∅Gap	Σveh	ΣCost ∅Gap	Σveh	ΣCost ∅Gap	Σveh	ΣCost ∅Gap
rm 1	8	73	21 831.5	73	21 849.0 *0.08*	73	21 830.8 *-0.01*	**73**	**21 817.6** *-0.06*	73	21 825.3 *-0.03*
rm 2	8	**16**	**21 477.3**	16	21 489.3 *0.05*	16	21 486.8 *0.04*	16	21 500.4 *0.11*	16	21 499.3 *0.10*
cm 1	8	86	25 821.1	84	25 754.3 *-0.30*	83	25 750.5 *-0.29*	83	25 609.3 *-0.85*	**83**	**25 550.6** *-1.08*
cm 2	8	37	33 249.9	37	33 265.4 *0.05*	37	33 319.6 *0.20*	37	33 279.0 *0.09*	**37**	**33 245.0** *-0.02*
rcm 1	8	**82**	**25 747.4**	82	25 805.2 *0.23*	82	25 815.7 *0.26*	82	25 801.7 *0.21*	82	25 806.3 *0.23*
rcm 2	8	**16**	**21 854.0**	16	21 865.0 *0.05*	16	21 881.1 *0.13*	16	21 881.1 *0.12*	16	21 878.4 *0.11*
Total	48	310	149 981.2	308	150 028.1 *0.03*	307	150 084.6 *0.06*	307	149 889.1 *-0.06*	307	**149 804.9** *-0.11*
# Best			18		17		16		20		**24**

Table 2.4 – Results for the VRPMTW benchmark of Belhaiza et al. (2014)

Table 2.4 shows aggregated results for the six groups of instances (R=random, C=clustered, or RC=partly random, partly clustered; series 1 and 2 with tight and wide time windows, respectively). Detailed results per instance are presented in Appendix A.5.

Comparing all five algorithms, the ALNS produces solutions with more vehicles than our SLNS configurations, which is caused by the different performance in the group cm 1. Regarding the overall-cost objective, results over all five algorithms are rather similar (total differences are below 0.2 %). SLNS+SPP+BS performs best, it improves the results of the ALNS by $0.11\,\%$ and provides the most best-known solutions. Over the 48 instances, 23 new best solutions are found by (at least) one of the four SLNS configurations. Note that the gap improvement is relatively small due to the large fixed cost for the utilization of vehicles.

2.5.4 Results for the VRPDO

For the VRPDO, we test the SLNS configurations on the instances described in Section 1.6. The required service levels are given by $\beta_1 = 80\,\%$ and $\beta_2 = 90\,\%$. The time budget is set to 300 seconds for 100-customers instances and 2 000 seconds for 200-customers instances.

Recall that the VRPDO has a hierarchical objective. Thus, Algorithm 2 is run twice, first to minimize the number of vehicles, and second to minimize the routing cost.

Table 2.5 presents the aggregated results. Each line refers to a group of instances with identical number of customers and options. Columns 3 and 4 indicate the total number of vehicles and the total cost of the best solutions found out of five runs of SLNS. For each of its variant we indicate the total cost and the average gap to the cost of the solutions of SLNS. [‡] indicates that one route less was needed. Detailed results can be found in Appendix A.1.

		SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS	
Group	#	Σveh	ΣCost	ΣCost	$\emptyset$Gap	ΣCost	$\emptyset$Gap	ΣCost	$\emptyset$Gap
U100	10	105	6 558.55	6 612.34	0.83	**6 528.21**	-0.42	6 548.26	-0.16
V100	10	104	7 078.08	7 070.62	-0.10	**7 026.62**	-0.71	7 052.34	-0.39
UBC100	10	40	**3 593.88**	3 657.44	1.75	3 599.04	0.16	3 612.59	0.58
U200	10	205	11 959.32	11 983.34	0.27	**11 768.85**	-1.51	11 786.57	-1.39
V200	10	204	13 716.05	13 725.32	0.06	13 741.99	0.29[‡]	**13 540.19**	-1.29
UBC200	10	80	6 062.81	6 227.54	2.76	6 072.03	0.18	6,232.32	2.74
Total		738	48 968.68	49 276.59	0.93	**48 736.74**	-0.34	48 772.27	0.01
# Best			16	7		**27**		**27**	

Table 2.5 – Results for the VRPDO benchmark of Dumez et al. (2021a)

Among the SLNS configurations, SLNS+SPP produces the best results with the smallest routing costs. Moreover, SLNS+SPP is the only configuration that is able to find a solution with one less vehicle on instance V_200_3. For all other instances the minimum number of vehicles is

identical over all configurations. SLNS+SPP improves the results compared to SLNS by 0.34% on average and by up to 1.51% on the U instances with 200 customers. It finds the best-known solution on 35 of the 60 instances.

Configuration SLNS+SPP+BS performs very similarly to SLNS+SPP. It finds two more best-known solution and the solution quality is less than 0.5% worse. Moreover, the detailed results of Appendix A.1 show that configuration SLNS+SPP+BS exclusively finds new best solutions for 14 instances.

2.5.5 Comparison of matheuristic configurations

To summarize the results for the four GVRPTW variants, we group all instances according to different criteria to further assess the performance of the matheuristic configurations. To this end, we report the gap to the best-known solution value and the number of best-known solutions found. To determine the best-known solution value, we include our own results from the previous tables. Formally, the gap (in percent) is computed as $100 \cdot (UB - BKS)/BKS$, where BKS is the best-known solution value from the literature, now including our results and UB is the best solution value found by the algorithm considered.

For the VRPDO, we consider only instances for which all four configurations achieve a solution with the same number of vehicles. As a consequence, the instance `V_200_3` is disregarded. We group the instances according to the VRP variant, the number of customers per route in the BKS, and the number of options per customer.

Table 2.6 shows aggregated results regarding the performance per group: The first column indicates the category used for grouping the instances, the second column gives the value defining the group, and the third column shows the number of instances in that group. The next four columns report the average gap and the last four columns report the number of best-known solutions found. In each line, the smallest average gap and the largest number of BKSs found are highlighted in bold.

Results grouped by		#	% Gap				# BKS			
			SLNS	SLNS+BS	SLNS+SPP	SLNS+SPP+BS	SLNS	SLNS+BS	SLNS+SPP	SLNS+SPP+BS
Variant	GVRPTW	20	0.27	0.42	**0.25**	0.27	**11**	9	10	**11**
	VRPRDL	60	0.01	0.04	**0.00**	0.01	59	55	**60**	59
	VRPHRDL	60	0.04	0.08	**0.03**	0.08	57	54	**58**	56
	VRPMTW	48	0.32	0.35	0.23	**0.18**	17	16	20	**24**
	VRPDO	59	1.28	2.24	**0.73**	1.31	16	7	**27**	**27**
Customers per Route	[1;6[	77	**0.00**	0.03	**0.00**	**0.00**	**77**	75	**77**	**77**
	[6;10[	83	0.37	0.44	0.21	**0.20**	50	44	56	**59**
	[10;20[	44	1.06	1.31	**0.41**	0.55	11	11	**24**	23
	[20;∞[	43	**0.50**	1.60	0.61	1.30	**18**	10	12	14
Options per Customer	[1;2[	32	0.95	0.95	0.46	**0.44**	4	9	11	**14**
	[2;3[	52	0.96	2.08	**0.65**	1.30	19	9	21	**22**
	[3;4[	97	0.04	0.09	0.03	**0.02**	90	83	**91**	**91**
	[4;∞[	66	0.22	0.27	**0.14**	0.19	43	39	**46**	**46**
Total		247	0.40	0.67	**0.25**	0.39	156	139	169	**173**

Table 2.6 – Comparison of the four matheuristic configurations

Regarding the different problem variants, all configurations except SLNS+BS perform rather similarly on the instances for the GVRPTW, VRPRDL, and VRPHRDL. For VRPMTW instances, SLNS+SPP+BS is clearly the best configuration, and for the VRPDO instances, SLNS+SPP is the best. In particular, regarding '# BKS', the two configurations with the SPP component outperform their counterparts without SPP component on the VRPMTW and VR-PDO.

The rather bad performance of the BS component on VRPDO instances can be attributed to the additional synchronizations constraints only present in this variant. These constraints imply that a larger set of resources has to be taken into account in the label-setting algorithm. As a result, the practical difficulty of the SPPRC increases substantially, making the BS component too time-consuming relative to the improvements obtained with BS.

When instances are grouped according to route length, we can clearly see that the two exact components (i.e., SLNS+SPP+BS) are beneficial for the route lengths between 6 and 20 customers. There is nearly no difference between configurations for routes with 6 or less customers, while the exact components worsen the results for more than 20 customers.

When grouped according to the number of options per request, it seems that gaps decrease when the number of options per customer rises. However, this trend is not clear-cut.

To summarize, configuration SLNS+BS performs worst, while both configurations SLNS+SPP and SLNS+SPP+BS are very competitive. The former produces the best gaps, while the latter provides the most BKS. Both configurations are also complementing each other well, because the overlap between the 169 and 173 BKS (see last two columns of Table 2.6) is only 145 instances. We can also conclude that the BS component is only beneficial in combination with the SPP component.

We are able to support the above interpretations with the help of *performance profiles* (Dolan and Moré, 2002). Performance profiles allow the comparison of a set $\mathcal{A}$ of algorithms which are all applied to the same benchmark set. We use the four configurations $\mathcal{A} = \{SLNS, SLNS + BS, SLNS + SPP, SLNS + SPP + BS\}$ as the algorithms to compare. The performance profile $\rho_A : [1, \infty) \rightarrow [0, 1]$ of configuration $A \in \mathcal{A}$ is a distribution function of the ratio z_A/z_{best}, where z_A is the solution value of configuration A and $z_{best} = \min_{A \in \mathcal{A}} z_A$. In other words, the value $\rho_A(\tau)$ is the fraction of instances for which algorithm A finds a solution within a factor of $\tau > 1$ of the cost of the considered solution relative to the cost of the best solution found by all algorithms in $\mathcal{A}$. In particular, $\rho_A(1)$ is the proportion of instances for which A has found the best solution among all matheuristic configurations.

We first compare performance profiles for all instances with those for groups of instances

(a) All instances

(b) Instances with $[6, 10)$ customers per route

(c) Instances with $[10, 20)$ customers per route

(d) Instances with $[20, \infty)$ customers per route
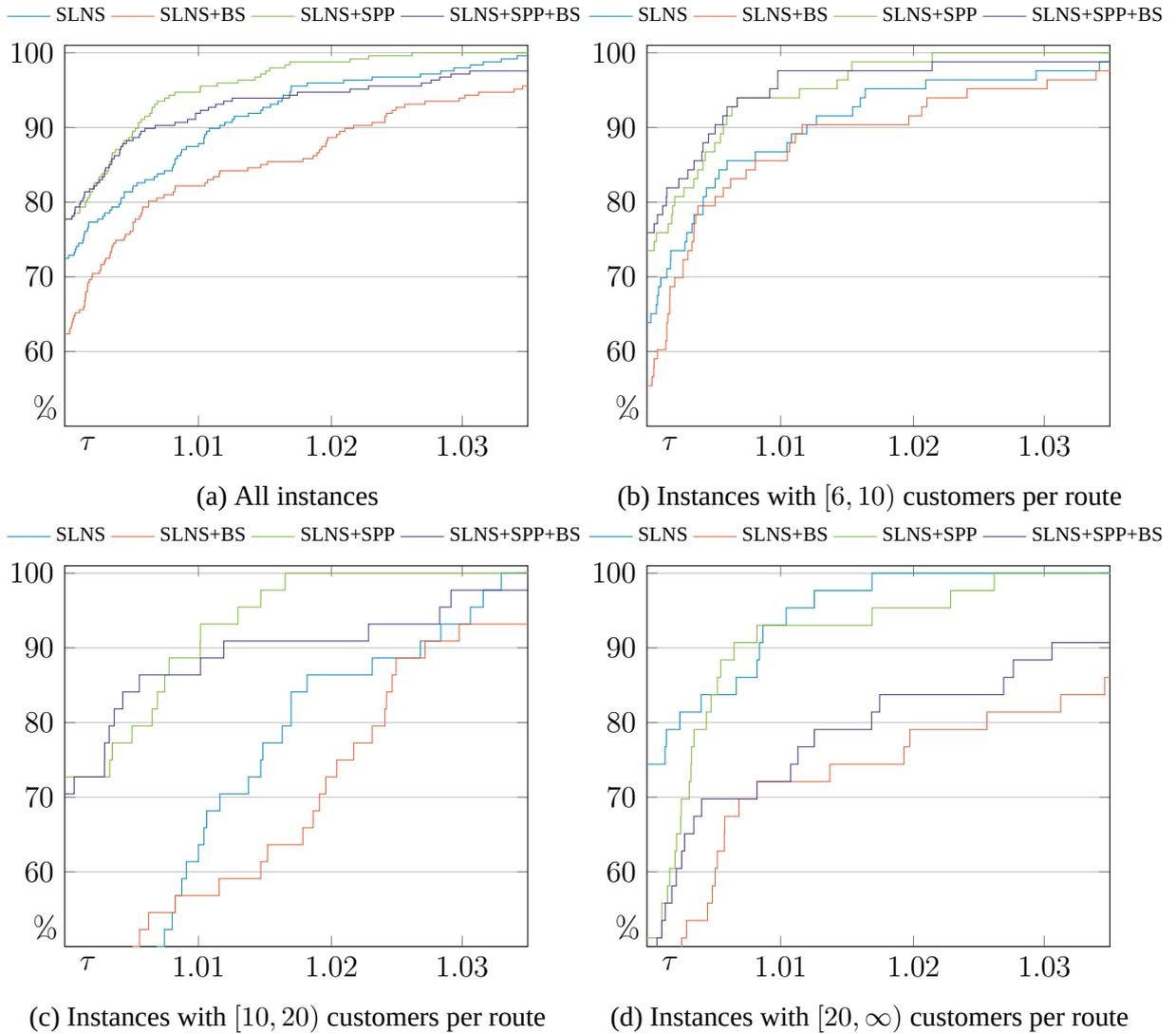
Figure 2.3 – Performance profile of the four variants, grouped by average route length

differing in the average number of customers per route (like the second section in Table 2.6). Figure 2.3 contains four performance profiles. Note that the group $[0, 6)$, i.e., with a very small number of average customers per route, has been omitted, since all but one configurations always find the BKS. The comparison of the profiles reveals:

— SLNS+BS is inferior to all other configurations in all groups.

— Comparing SLNS+SPP and SLNS+SPP+BS, we can state that the Balas-Simonetti neighborhood is only providing a small advantage for the group $[6, 10)$ with rather short routes. In the group $[10, 20)$, the configuration SLNS+SPP+BS using BS neighborhood is complementary to the configuration SLNS+SPP (note the overlapping profiles).

— The longer the routes, the more obvious and pronounced is the difference between SLNS and SLNS+BS as well as between SLNS+SPP and SLNS+SPP+BS.

— For the groups $[6, 10)$ and $[10, 20)$, it is evident that the set-partitioning component in SLNS+SPP is indispensable to reach the high-quality solutions.

We also pointed out that the performance of the BS component strongly depends on the number of resources to be included in the SPPRC. Recall that additional resources are needed in the VRPDO to count the number of shipments delivered to every shared location and for counting priorities. Hence, a deterioration of the BS component in the VRPDO can be expected when compared to the other GVRPTW variants.
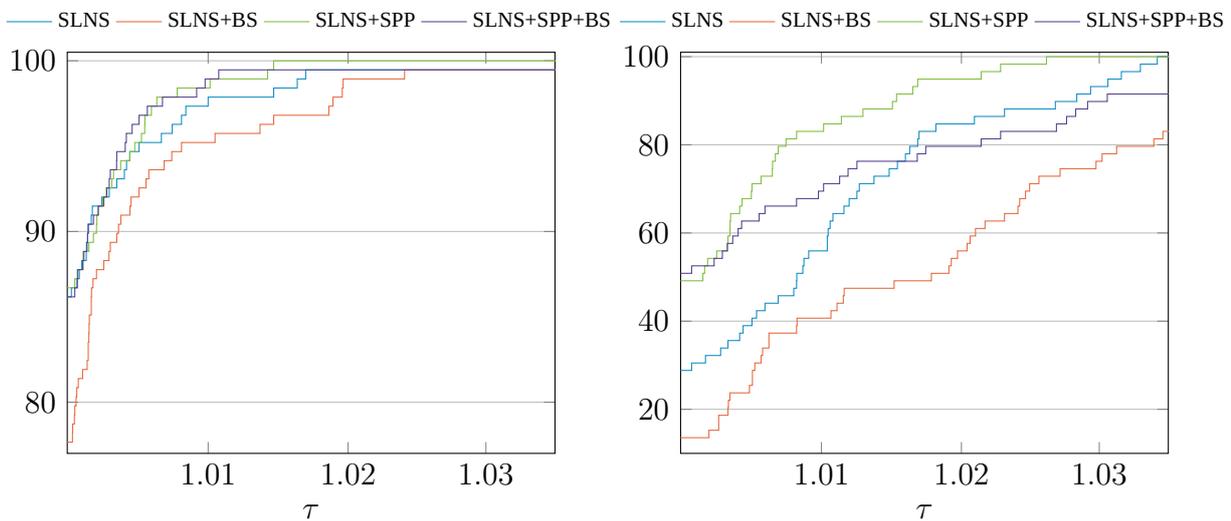
(a) All GVRPTW except VRPDO instances   (b) Only VRPDO instances

Figure 2.4 – Performance profiles of the solution value relative to the best found solution grouped by GVRPTW variant

The performance profiles shown in Figure 2.4 strongly support the given interpretation.

— Differences between SLNS and SLNS+BS as well as between SLNS+SPP and SLNS+SPP+BS are small for non-VRPDO instances but substantial in the VRPDO case.

— The profiles of SLNS+SPP and SLNS+SPP+BS are almost identical for non-VRPDO instances (Figure 2.4a). This is in line with the observation that the two configurations are complementing each other. As observed for the VRPMTW, the BS component in SLNS+SPP+BS can even lead to a superior performance.

— The comparison between Figure 2.4a and 2.4b shows that all configurations perform inferior on VRPDO instances. This is a strong indication that this GVRPTW variant is more 'difficult'.

— For values of τ very close to 1, i.e., for finding high-quality solutions, the BS component is generally useful.

— Also here the set-partitioning component is indispensable to reach the high-quality solutions, in particular for the VRPDO.

2.5.6 Additional experiments

In addition to the results published in Dumez et al. (2021c), we would like to present some latest results.

They were performed on Ubuntu 20.04 LTS running on an Intel Xeon Gold 6230 CPU @ 2.10GHz with IBM Ilog CPLEX 20.1. The computing time was scaled down by a factor 1.7 according to PassMark-Software (2020).

VRPRDL and VRPHRDL

Very recently, Yuan et al. (2021) also solved these VRP(H)RDL instances with an heuristic method. These solutions have a fleet size corresponding to the number of vehicles used in Ozbaygin et al. (2017). We compare the solutions of these two studies with those of SLNS+SPP with the same number of routes. Aggregated results are presented in Table 2.7 for the VRPRDL and in Table 2.8 for the VRPHRDL. Each line corresponds to an instance group specified by the first column. The second column indicates the number of instances in the group and the third column, the number of customers. Columns 4 and 5 detail the results provided by Ozbaygin et al. (2017): the total number of routes and the total cost of their solutions. Columns 6 and 7 show the results of the Column-Generation Based Heuristic (CGBH, Yuan (2019); Yuan et al. (2021)): the total cost of their solutions and the total computing time. Columns 6 and 7 give the same information for our SLNS+SPP.

First of all, let us underline that the results of the CGBH and those of SLNS+SPP are extremely close. Over all the instances of both problems, the largest relative deviation in the cost of the solutions from both algorithms is of 0.81%. On average, the deviation between our method results and those of Yuan et al. (2021) is 0.01% on the VRPRDL, and 0.02% on the VRPHRDL. Thus, both are improving the results of Ozbaygin et al. (2017).

Second, about the computing time, CGBH is faster on smaller instances but SLNS+SPP is faster on the larger ones. Nonetheless, for these experiments, we used a processor 1.4 faster than their (PassMark-Software, 2020).

Instances	#	$|N|$	BCP		CGBH		SLNS+SPP	
			Σveh	Σcost	Σcost	ΣTime	Σcost	ΣTime
1-5	5	15	24	**6 072**	**6 072**	1	**6 072**	15
6-10	5	20	28	**6 848**	**6 848**	2	**6 848**	20
11-20	10	30	68	**18 595**	**18 595**	10	**18 595**	50
21-30	10	60	129	**37 213**	**37 213**	44	**37 213**	70
31-40	10	120	195	53 881	53 768	621	**53 738**	350
41-50_v1	10	40	94	29 842	29 842	22	**29 838**	60
41-50_v2	10	40	75	**21 863**	**21 863**	43	**21 863**	60
Total	60		613	174 314	174 201	743	**174 167**	625

Table 2.7 – Comparison to Yuan et al. (2021) on the VRPRDL

Instances	#	$|N|$	BCP		CGBH		SLNS+SPP	
			Σveh	Σcost	Σcost	ΣTime	Σcost	ΣTime
1-5	5	15	19	**5 450**	**5 450**	1	**5 450**	15
6-10	5	20	20	**5 604**	**5 604**	3	**5 604**	20
11-20	10	30	52	**15 128**	**15 128**	20	**15 128**	50
21-30	10	60	83	26 829	26 801	163	**26 800**	70
31-40	10	120	132	38 610	37 296	2 445	**37 234**	350
Total	60		306	91 621	90 276	2 634	**90 216**	625

Table 2.8 – Comparison to Yuan et al. (2021) on the VRPHRDL

GVRPTW

In addition to results on the VRP(H)RDL, Yuan et al. (2021) also present results on the GVRPTW instances of Moccia et al. (2012). Their method, CGHB, slightly reduces the average cost of the solutions and significantly reduces the computing time compared to the results of Moccia et al. (2012).

Table 2.9 details the best results obtained out of five runs by SLNS+SPP, compared with the results of CGHB. Note that the authors did not indicate whether their results are the best

results out of several runs. For each instance, the cost of the solution and the computing time (in second) are reported for each of the two methods.

The two methods provide very similar results on this dataset. Indeed, the average gap between the cost of the solutions is only 0.06%. However, the computing time of SLNS+SPP is, in average, 3.5 times smaller, with a processor 1.4 faster than theirs (PassMark-Software, 2020).

	CGHB		SLNS+SPP	
Instance	Cost	Time	Cost	Time
i-030-04-08	3 498	8.74	**3 497**	5
i-030-08-12	**2 796**	15.37	**2 796**	5
i-040-04-08	**3 811**	6.46	**3 811**	8
i-040-08-12	**3 768**	7.80	**3 768**	8
i-050-04-08	**5 439**	10.08	5 447	11
i-050-08-12	4 054	12.35	**4 034**	11
i-060-04-08	**5 908**	49.10	5 926	14
i-060-08-12	**4 303**	26.26	**4 303**	14
i-070-04-08	**6 228**	24.78	6 246	22
i-070-08-12	4 694	45.27	**4 644**	22
i-080-04-08	7 420	179.33	**7 394**	27
i-080-08-12	**5 613**	43.65	5 661	27
i-090-04-08	**7 108**	328.34	7 182	32
i-090-08-12	5 893	307.66	**5 869**	32
i-100-04-08	7 339	85.78	**7 308**	38
i-100-08-12	6 788	419.38	**6 546**	38
i-110-04-08	**8 618**	128.12	8 687	45
i-110-08-12	**6 343**	120.46	6 487	45
i-120-04-08	8 455	190.67	**8 357**	60
i-120-08-12	**6 772**	151.67	6 829	60
Total	114 848	2 194.66	114 792	524

Table 2.9 – Comparison to Yuan et al. (2021) on the GVRPTW

Large VRPDO instances

Likewise, since the submission of the previously presented results, we ran experiments on VRPDO instances with 400 customers.

Table 2.10 presents the aggregated results of each variant of the developed method on these large instances. The best solutions out of five runs with 3500 seconds of time budget were taken

into account. Each line correspond to an instance type. For each configuration of SLNS the table shows the total number of vehicles and the total cost of the solutions of each type.

Overall the SPP improves the results of SLNS by 1.1%, while the Balas-Simonetti worsen them by 2.4%, and the conjunction of the two worsen the results by 0.6%.

The tendencies observed on the other instances are confirmed, and even amplified, on these larger instances. That is to say, SLNS provides good results overall. SLNS+SPP is able to slightly improve the results. And SLNS+SPP+BS is able to find better solutions on some very specific instances (it obtains a unique best solution on 2 instances). However, on average, the dynamic programming component is too time consuming on these large instances.

Instances	SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS	
	Σveh	Σcost	Σveh	Σcost	Σveh	Σcost	Σveh	Σcost
U 400	406	22 836.55	406	23 334.82	406	**22 358.59**	406	22 856.93
V 400	405	22 554.79	405	22 884.42	405	**22 342.08**	405	22 470.25
UBC 400	156	**11 223.36**	156	11 756.99	156	11 288.26	156	11 629.41
Total	967	56 614.70	967	57 976.23	967	**55 988.93**	967	56 956.59

Table 2.10 – Results for the VRPDO instances with 400 customers

2.6 Conclusions

In this chapter, we presented a SLNS-based matheuristic that can cope with GVRPTW variants, in particular also with the most general and more involved VRPDO. The matheuristic has an adaptive layer that controls the use of two (optional) exact components. The Balas-Simonetti neighborhood allows an improvement of solutions when the standard SLNS process stalls. The Set Partionning Problem component utilizes a MIP solver to selects routes from a larger pool of potential routes. The second component uses BS neighborhood that we adapt for the use in a multiple vehicle context with capacity, time window, and inter-route constraints. Both components are, compared to a single SLNS iteration, very time-consuming. Hence, the adaptive layer very carefully controls how often and when the exact components are invoked.

The primary focus of our research is the evaluation of the adaptive layer as well as the two exact components. We compare four different configurations of the new matheuristic (SLNS, SLNS+BS, SLNS+SPP, and SLNS+SPP+BS, i.e., with and without SPP and BS components) among each other and with state-of-the-art algorithms for the GVRPTW, VRPRDL, VRPHRDL, VRPMTW, and VRPDO. The experiments are conducted on standard benchmarks for these problems and lead to the following insights:

— The two configurations with the SPP component outperform their counterparts without SPP component regarding average gaps to best-known solutions (BKS) and the number of BKS computed (Table 2.6).

— The configuration with the BS component alone (SLNS+BS) is inferior. In particular, for the VRPDO, the presence of many inter-route constraints that must be handled within the BS neighborhood exploration makes the BS component too time-consuming (see Section 2.5.4). However, when combined with SPP the resulting configuration SLNS+SPP+BS is competitive.

— The practical difficulty of the VRPDO is well reflected in the results delivered by all matheuristic configurations. Average gaps of 0.73 % for the VRPDO are much bigger than gaps for the other GVRPTW variants which fall below 0.25 % (Table 2.6).

— Compared to state-of-the-art metaheuristics from the literature, the new matheuristic is much faster (factor 7 for the GVRPTW; factor 2 for the VRPMTW, see Sections 2.5.1 and 2.5.3).

— For the VRPRDL (VRPHRDL), the comparison against a recent exact branch-price-and-cut algorithm shows that configuration SLNS+SPP finds all (all except one) known optimal solutions in a fraction of the computation time available for the exact algorithm. In

addition, all configurations together provide new best-known solutions for all instances that were not solved to proven optimality by the branch-price-and-cut algorithm (Section 2.5.2).

Thus, SLNS is an efficient method for generalized vehicle routing problems with time windows. Hybridizing it with the SPP enhances its performances by a few percent, especially on hard instances. Finally, the Balas-Simonetti can further improve these results, but only on instances with short routes.

Furthermore, to be efficient on large instances, it is needed to speed-up the Balas-Simonetti neighborhood for problems with synchronized resources. We can think about the pulse algorithm (Lozano and Medaglia, 2013). In addition, the computing time of all the configurations also suggests to reduce the number of evaluated insertion positions for each customer, like proposed by Toth and Vigo (2003).

AN ITERATIVE TWO-STAGE HEURISTIC FOR THE 2-ECHELON MULTI-TRIP VEHICLE ROUTING PROBLEM WITH CAPACITATED SATELLITES AND REVERSE FLOWS

3.1   Introduction

In more and more cities, governmental regulations restrict the number and type of vehicles allowed to access city centers because of the intense inner-city traffic, narrow streets, and the lack of adequate parking spaces (Muñuzuri et al., 2013). Coordinated urban deliveries can be part of the solution. We consider modern inner-city distributions systems that rely on a multi-echelon structure to reduce the nuisances associated with freight transportation in urban areas while supporting their economic and social development (Crainic et al., 2009).

One of the most common designs for handling and reducing the large number of freight vehicles going into cities, often delivering small quantities per stop, is to consolidate this fragmented volume at *urban distribution centers* (UDCs) located in cities outskirts (Savelsbergh and Van Woensel, 2016). In the *two-echelon vehicle routing problem* (2E-VRP), customers are supplied from the UDC through intermediary *satellites*. First-echelon vehicles transport goods from the UDC to satellites, at which second-echelon vehicles collect the goods and deliver them to the customers. The flow of goods must respect operation and load synchronization constraints (following the taxonomy of Drexl, 2012), i.e., one must decide on the assignment of each customer's demand to a satellite and ensure that incoming and outgoing quantities at each satellite coincide.

We have identified three extensions of the basic 2E-VRP as highly relevant. These are: (i) the integration of reverse flows, (ii) the consideration of several timing aspects (including time windows and the possibility to perform multiple trips with the same vehicle), and (iii) satellite capacities. We elaborate their importance in the following.

The goal to reduce traffic, pollution, and noise in city centers certainly requires a holistic view on the logistics system including goods transportation into as well as out of the city. Bektaş et al. (2017) pointed out that handling forward, reverse, and transiting flows is a key activity within urban areas. In some fields of application, the consideration of forward and reverse flows is therefore almost imperative. Examples include beverage-delivery services (Bruck and Iori, 2017) and package and urban courier services (Wong, 2008), where reverse quantities of some customers can easily exceed their forward quantities.

Typical fleets on the second echelon comprise small trucks (vans, Sprinters) as well as locally emission-free vehicles such as *battery electric vehicles* (BEV, Desaulniers et al., 2016; Schneider et al., 2014) and cargo bikes (Elbert and Friedrich, 2020). These vehicles typically have a much smaller capacity than the trucks used on the first echelon. As a result, second-

echelon routes are relatively short so that a vehicle can be cleared and replenished several times. With a limited fleet, the associated problem is therefore a *Multi-Trip VRP* (Paradiso et al., 2020, stress its relevance for city logistics and last-mile delivery). Moreover, when distances between UDC and satellites are short, multiple trips of the first-echelon vehicles are also possible (see, e.g., Nolz et al., 2020).

In their review, Cuda et al. (2015) highlight the steadily increasing number of publications on the 2E-VRP. They conclude that mostly, basic versions of the 2E-VRP have been studied and that many practical issues, in particular temporal interdependencies are still to be further investigated both in heuristic and exact approaches. Temporal interdependencies may refer to the requirement that:

(1) either first-echelon and second-echelon vehicles have to meet at the satellite in order to transfer loads (no storage, *exact operation synchronization* as defined by Drexl, 2012)

(2) or satellite visits of first-echelon vehicles can precede those of second-echelon vehicles (storage at satellites, *operation synchronization with precedences*, see also Drexl, 2012).

Furthermore, satellite capacities impose *resource synchronization* constraints (Drexl, 2012).

For case (1), Grangier et al. (2016) introduce the *two-echelon multiple-trip vehicle routing problem with satellite synchronization* (2E-MTVRP-SS) which extends the basic variant by customer time windows and multiple trips on the second echelon. In their variant, there are no storage possibilities at the satellites. For case (2), it is obvious that modeling and solving problems with satellite capacities is much more involved. Only very few works cover this aspect (see literature review in Section 3.2).

In response to this research gap, we aim at modeling and solving 2E-VRP with time-windows with a specific focus on incorporating the processes at the satellites in combination with the requirement to handle both forward and reverse flows (goods and returns). Overall, we extend the 2E-MTVRP-SS of Grangier et al. (2016) by three aspects:

— First, customers can have a pickup demand introducing reverse flows into the two-echelon system. As a consequence, second-echelon vehicles need to simultaneously deliver goods and collect pick-up goods at customers within their specified time window. The pickup demand must be transported to a satellite and from this satellite to the UDC by a first-echelon vehicle.

— Second, also first-echelon vehicles are allowed to perform multiple trips during the planning horizon.

— Third, satellites allow a temporary storage of goods, but they feature a limited capacity in

terms of goods that can be stored at a time. The most prevalent situation is probably that goods to be delivered (forward) and picked-up (reverse) are stored together, thus, they share and compete for the satellite capacity.

We introduce the *Two-Echelon Multiple-Trip Vehicle Routing Problem with Capacitated Satellites and Reverse Flows* (2E-MTVRP-CSRF) that addresses all three extensions together. The 2E-MTVRP-CSRF models the time and quantity dependencies at capacitated satellites in a more precise manner than previous works. It is, therefore, obvious that for 2E-MTVRP-CSRF instances of realistic size, the only viable solution approach is heuristic.

To solve the 2E-MTVRP-CSRF, we propose an iterative algorithm that partially decomposes the problem into three different subproblems. For two of them, we present two *Large Neighborhood Search* (LNS) algorithms, each of them optimizing one echelon while parts of the other echelon are fixed. With one sub-problem per echelon, the method takes advantage of the effectiveness of LNS to solve Multi-Trip Vehicle Routing Problem with Reverse Flow and to integrate synchronization constraint with the other echelon. For the third subproblem, an enhanced version of the path-based formulation is used to recombine trips found in both LNS algorithms. This way, the method also takes advantage of the MIP-based approach to recombine trips generated by the two LNSs, while having a global view on satellite-capacity constraints as well as the spatial and time synchronization between first and second echelon.

On randomly generated instances with 100 customers and 4 satellites of medium capacity, we find that the overhead cost of considering the satellites capacity is of 6.5%, which is worth considering in regard of the meter square prices in large cities. In addition, on those instances, we computed that integrating forward and reverse flow reduces the routing costs by up to 40%.

The remainder of the paper is structured as follows. Section 3.2 reviews the 2E-VRP literature focusing on those variants with reverse flows, timing aspects including multiple trips, and satellite capacities. In Section 3.3, we formally introduce the 2E-MTVRP-CSRF, detail feasibility conditions, and present a trip-based formulation. Section 3.4 presents the matheuristic algorithm. Computational results and managerial insights are presented in Section 3.5, before final conclusions are drawn in Section 3.6.

3.2 Literature

The basic 2E-VRP implicitly assumes that all transfers between the first and second echelon happen in separate periods and are therefore always operable, e.g., assuming a fixed cut-off period during which all transfers are performed so that all first(second)-echelon vehicles must arrive (depart) before (after) that period. Hence, operation synchronization is *pure spatial* in the basic version (see Drexl, 2012). The first study on the 2E-VRP was conducted by Gonzalez-Feliu et al. (2008) who introduced the problem into the literature. They proposed a commodity-flow model and a *branch-and-cut* (BC) algorithm that was tested on instances with up to 50 customers and 4 satellites. The 2E-VRP was surveyed by Cuda et al. (2015). More recent successful solution approaches are exact (Marques et al., 2020b; Perboli et al., 2018) and heuristic (Amarouche et al., 2018; Breunig et al., 2016; Wang et al., 2017).

The standard assumption is that customers are served by a single visit so that demands are not split on the second echelon. However, splitting customer demands on the first echelon is an option.

3.2.1 Reverse flows

The integration of forward and reverse flows in a two-echelon distribution system was presented as an important challenge Crainic et al. (2012). Belgin et al. (2018) were the first to solve a 2E-VRP variant with forward and reverse flows so that a customer can have both a delivery and pickup demand. The vehicle routing problem at the second echelon is one with *simultaneous deliveries and pickups* (Subramanian et al., 2013b). Additionally, Belgin et al. model *satellite capacity* as the maximum total demand that can be served by the satellite during the planning horizon (we also use the term *total demand* in the following to refer to delivery plus pickup demand). The problem is solved heuristically by combining *Variable Neighborhood Descent* (VND, Hansen et al., 2009) and local search. Additionally, they present a two-index model and computationally evaluate the effect of additional valid inequalities on lower bounds. Tests were conducted on a new instance set comprising up to 50 customers and 5 satellites.

3.2.2 Time windows and multiple trips

Crainic et al. (2009) were the first to consider timing aspects in 2E-VRP. The authors discuss customer time windows and multiple trips on both echelons among other features of a general

two-tier city logistic systems. The work also presents several modeling ideas and describes possible heuristics without computational testing.

Grangier et al. (2016) introduced the 2E-MTVRP-SS which features customer time windows and multiple trips in the second echelon. They present a LNS algorithm with a fast and sophisticated feasibility check. Computational experiments were conducted on instances with up to 8 satellites and 100 customers derived from the VRPTW benchmark instances of Solomon and Desrosiers (1988).

Anderluh et al. (2017) extend the 2E-MTVRP-SS by allowing the first-echelon vehicle to serve some customers. A two-phase greedy randomized adaptive search procedure with path relinking is applied to solve the extended 2E-MTVRP-SS. Computational tests were conducted on instances with up to 125 customers and 18 satellites.

Dellaert et al. (2019) suggest a two-path formulation for the single-trip multi-depot 2E-VRP with time windows and exact synchronization (2E-VRPTW). They assume that the entire demand delivered by a second-echelon vehicle must be provided by a single first-echelon trip. A *branch-and-price* (BP) algorithm that enumerates configurations of first-echelon trips is a key component. The BP algorithm is capable to exactly solve some instances with up to 100 customers and 5 satellites within the 3-hour time limit. Mhamedi et al. (2020) developed a sophisticated *branch-price-and-cut* (BPC) algorithm for the 2E-VRPTW. The algorithm performs well on the instances generated by (Dellaert et al., 2019), computing 41 new optimal solutions in the 3-hour time limit. Independently, Marques et al. (2020a) present BPC algorithms to solve the 2E-MTVRP-SS and 2E-VRPTW. They reported new optimal solutions for 9 of the 2E-MTVRP-SS instances and 54 of the 2E-VRPTW instances.

The practical relevance of timing aspects has led to the introduction of several other 2E-VRP variants covering new technologies and real-world features such as electric vehicles (Jie et al., 2019), cargo bikes with swap containers (Mühlbauer and Fontaine, 2021), and multiple commodities provided by different UDCs (Dellaert et al., 2021). Recently, Anderluh et al. (2020) analyzed and discussed the impact of stochastic travel times on costs.

3.2.3   Satellite capacity

Two approaches for modelling the satellite capacities are common in the literature. In the first approach, an upper bound on the number of second-echelon vehicles leaving the satellite is imposed (first suggested by Crainic et al. (2009) and very recently employed by Mühlbauer and Fontaine (2021)). The second approach limits the total demand that can be served by a satellite (in the uni-demand case, this means limiting the number of served customers, see, e.g., Belgin

et al., 2018; Breunig et al., 2016). Both approaches have in common that they do not consider the dynamics over the planning horizon, i.e., available or residual capacities depending on the goods entering and leaving the satellite over time.

To the best of our knowledge, only three works use a more detailed time-dependent capacity model. Li et al. (2018) study a 2E-VRP with customer time windows, a given assignment of customers to satellites, and real-time transshipment capacities. The real-time transshipment capacities describe the currently available capacity of the satellite defined as the difference between the maximum capacity and the quantity of goods currently waiting at a satellite to get picked-up by a second-echelon vehicle. Additionally, the problem features split deliveries on the first echelon and multiple trips on the second echelon. A MIP formulation is provided and a two-stage heuristic based on *variable neighborhood search* (VNS, Hansen et al., 2009) is proposed. Small-scale instances with up to 3 satellites and 5 customers per satellite were solved to optimality with a MIP solver within a 4-hours time limit. The two-stage heuristic has been tested on 99 larger instances with up to 30 satellites and 30 customers per satellite. Li et al. (2020) slightly extend the problem by varying the maximum transshipment capacity for each time period.

Recently, Nolz et al. (2020) considered a two-echelon distribution with a single capacitated satellite, customer time windows, and multiple trips on both echelons. They propose a new MIP model and a three-phase heuristic method which uses population-based meta-heuristics and integer programs. Nolz et al. evaluate their methods on instances with up to 81 customers generated from real-world data from the city of Vienna.

More generally, a time dependent satellite capacity involves synchronizing routes on shared renewable capacitated resources at facilities. To our knowledge, this type of problem has received little attention in the literature, most resource constrained routing and scheduling problems coming from the sharing of particular vehicles or personnel with different skills that need to be routed to perform different tasks (Castillo-Salazar et al., 2016; Fikar and Hirsch, 2017; Paraskevopoulos et al., 2017). Several cases are related to loading or unloading unary ressources in forestry (El Hachemi et al., 2011, 2013), construction (Schmid et al., 2009) or public work (Grimault et al., 2017). In Grangier et al. (2019), a constraint limit the number of dock that can be used simultaneously in the VRP with cross-docking. Froger et al. (2017) integrate a limited number of chargers at charging stations in electric VRPs. In the two later contributions, each vehicle consumes one unit of the resource capacity for a given time when it uses it. But to our knowledge, no paper investigate a synchronization constraint on a renewable resource that is expressed on a transferred quantity, in particular in a problem with forward and reverse flows.

3.3 The 2-Echelon Multi-Trip Vehicle Routing Problem with Capacitated Satellites and Reverse Flow

We define the 2E-MTVRP-CSRF by formally describing customers, satellites, fleet, trips, routes, and then satellite capacities. The three latter aspects require a detailed explanation provided in Sections 3.3.1, 3.3.2 and 3.3.3. An integer (linear) programming (IP) model is then presented in Section 3.3.4.

Let N denote the set of customers, where each customer $i \in N$ has a delivery (=forward or linehaul) demand q_i^{fw} and a pick-up (=reverse or backhaul) demand q_i^{rv}. Both demands have to be fulfilled by a single visit of a second-echelon vehicle. For the visit of customer $i \in N$, a vehicle may wait at the customer before actually providing service, which must start within the time-window $[a_i, b_i]$.

Let S denote the set of satellites. We assume that satellites $s \in S$ are available over the entire planning horizon $\mathbb{T} = \{0, \ldots, \bar{t}\}$, i.e., spanning the time window $[a_s, b_s] = [0, \bar{t}]$. Moreover, each satellite $s \in S$ has a limited capacity C_s^{sat} and a constant processing time p_s for transferring goods from one echelon to the other. How the capacity and processing time have to be interpreted is detailed in Section 3.3.3.

A homogeneous fleet F^1 of first-echelon vehicles is stationed at the UDC o^1. Likewise, a homogeneous fleet F^2 of second-echelon vehicles is hosted at the second-echelon depot o^2. All vehicles must start and end their routes at their depots o^1 and o^2, respectively. They are allowed to perform multiple trips in between. Each first(second)-echelon vehicle has a capacity of Q^1 (Q^2) that has to be shared by forward and reverse demands when transported together.

The objective of the 2E-MTVRP-CSRF is to find cost-minimal sets of feasible first-echelon and second-echelon trips (formally defined in Section 3.3.1) such that the following constraints hold:

(F1) All customers are visited exactly once by exactly one second-echelon trip.

(F2) The set of first(second)-echelon trips can be combined to at most $|F^1|$ ($|F^2|$) feasible first(second)-echelon routes.

(F3) The forward flow of goods is conserved at each satellite $s \in S$ and each point in time $t \in \mathbb{T}$, i.e., the forward flow that has reached satellite s by first-echelon trips until time $t - p_s$ is at least as large as the forward demands leaving satellite s on second-echelon trips until time t.

(F4) Likewise, the reverse flow of goods is conserved at each satellite $s \in S$ and each point

in time $t \in \mathbb{T}$, i.e., the collected demands reaching satellite s on second-echelon trips arriving until time t is at least as large as the reverse flow that has left from the satellite s by first-echelon trips starting until time $t + p_s$.

(F5) The capacities of all satellites are never exceeded, i.e., at any point in time, the quantity of goods stored or processed at a satellite is not larger than the satellites capacity.

Note that flow conservation via (F3) and (F4) allows a customer's demand (either forward or reverse demand) to be split on the first-echelon using two or more trips (with possibly different vehicles), but that this demand cannot be split on the second echelon.

3.3.1 Trips

Next, we define feasible trips and routes in both echelons with the help of two directed graphs. For the first echelon, let $G^1 = (V^1, A^1)$ be the complete digraph with vertex set $V^1 = S \cup \{o^1\}$ and arc set A^1. Each arc $(i, j) \in A^1$ is associated with a non-negative travel time t_{ij}^1 and a non-negative travel cost c_{ij}^1.

A *first-echelon trip* $h = (P, T, L)$ consists of a closed directed walk $P = (i_0, i_1, \ldots, i_n, i_{n+1})$ with $i_0 = i_{n+1} = o^1$ and $i_1, \ldots, i_n \in S$, a *time schedule* $T = (T_0, T_1, \ldots, T_n, T_{n+1}) \in \mathbb{T}^{n+2}$, and a *loading plan* $((L_1^{fw}, L_1^{rv}), \ldots, (L_n^{fw}, L_n^{rv})) \in ([0, Q^1] \cap \mathbb{Z})^{2 \times n}$. The attribute $T_k \in \mathbb{T}$ models the start of the service/operation at vertex i_k for $k \in \{0, 1, \ldots, n+1\}$.

Each pair (i_k, T_k) of vertex and visit time is called *First-Echelon Visit* (FEV). Note that multiple FEVs at the same satellite are allowed within a walk, including consecutive operations happening at the same satellite at different times (these loops may model consecutive appointments at the satellite, for example). Finally, the attributes L_k^{fw} and L_k^{rv} are the quantities dropped-off and collected, respectively, at satellite i_k for $1 \leq k \leq n$.

A first-echelon trip $h = (P, T, L)$ is *feasible* if the following three conditions hold:

(Tr1) all vertices are visited within the planning horizon:

$$T_k \in [0, t^{\max}] \qquad \text{for all } k \in \{0, 1, \ldots, n+1\},$$

(Tr2) the time schedule is consistent with respect to travel and processing times:

$$T_k + t_{i_k, i_{k+1}}^1 + p_{i_k} \leq T_{k+1} \qquad \text{for all } k \in \{0, 1, \ldots, n\},$$

(Tr3) and the load at each vertex is less than the capacity of a first-level truck:

$$\sum_{k=l+1}^{n} L_k^{\text{fw}} + \sum_{k=1}^{l} L_k^{\text{rv}} \le Q^1 \qquad \text{for all } l \in \{0, 1, \dots, n\},$$

where we assume a processing time $p_{o^1} = 0$ for the UDC. The *cost* c_h^1 of the first-echelon trip h is given by $\sum_{k=0}^{n} c_{i_k, i_{k+1}}^1$.

Similarly, let $G^2 = (V^2, A^2)$ be the simple and complete digraph with vertex set $V^2 = S \cup N \cup \{o^2\}$ and arc set A^2. Each arc $(i, j) \in A^2$ is associated with a non-negative travel time t_{ij}^2 and a non-negative travel cost c_{ij}^2.

A *second-echelon trip* $h = (P, T)$ comprises a (closed or open) walk $P = (i_0, i_1, \dots, i_n, i_{n+1})$ in G^2 with $i_0, i_{n+1} \in S \cup \{o^2\}$ and $i_1, \dots, i_n \in N$, and a time schedule $(T_0, \dots, T_{n+1}) \in \mathbb{T}^{n+2}$.

The second-echelon trip $h = (P, T)$ is *feasible* if the following five conditions hold:

(Tr4) the vertices are visited within their time windows:

$$T_k \in [a_{i_k}, b_{i_k}] \qquad \text{for all } k \in \{0, 1, \dots n+1\},$$

(Tr5) the time schedule respects travel times:

$$T_k + t_{i_k, i_{k+1}}^2 \le T_{k+1} \qquad \text{for all } k \in \{0, 1, \dots, n\},$$

(Tr6) the vehicle capacity is respected at each vertex on the trip:

$$\sum_{k=l+1}^{n} q_{i_k}^{\text{fw}} + \sum_{k=1}^{l} q_{i_k}^{\text{rv}} \le Q^2 \qquad \text{for all } l \in \{0, 1, \dots n\},$$

(Tr7) if $i_0 = o^2$, then $q_{i_k}^{\text{fw}} = 0$ must hold for all $k \in \{1, 2, \dots, n\}$, and

(Tr8) if $i_{n+1} = o^2$, then $q_{i_k}^{\text{rv}} = 0$ must hold for all $k \in \{1, 2, \dots, n\}$.

The latter two conditions (Tr7) and (Tr8) impose that trips that start (end) at the depot o^2 cannot serve customers that have a positive delivery (reverse) demand. The *cost* c_h^2 of the second-echelon trip h is given by $c_h^2 = \sum_{k=0}^{n} c_{i_k, i_{k+1}}^2$.

Note that second-echelon vehicles and trips can start and end at different satellites. In particular, second-echelon trips that do not visit any customers are allowed (i.e., $n = 0$). These

transfer trips model that a vehicle can change the starting satellite for its next trip. In addition, service times at customers are integrated into travelling times.

3.3.2 Routes

A *first(second)-echelon route* R is a sequence $(h_1, \ldots, h_m)$ of $m \geq 1$ feasible first(second)-echelon trips h_j. We denote the walks by $P_j = (i_{j0}, i_{j1}, \ldots, i_{j,n_j}, i_{j,n_j+1})$ and the time schedules by $T_j = (T_{j0}, T_{j1}, \ldots, T_{j,n_j}, T_{j,n_j+1})$ for $j \in \{1, 2, \ldots, m\}$.

A first-echelon route is *feasible* if

(R1) two consecutive trips h_j and h_{j+1} do not overlap in time:

$$T_{j,n_j+1} \leq T_{j+1,0} \qquad \text{for all } j \in \{1, \ldots, m-1\}.$$

A second-echelon route is *feasible* if the following three conditions hold:

(R2) it starts and end at the second-echelon depot:

$$o^2 = i_{1,0} = i_{m,n_m+1},$$

(R3) consecutive trips share identical end and start satellites:

$$i_{j,n_j+1} = i_{j+1,0} \in S \qquad \text{for all } j \in \{1, 2, \ldots, m-1\},$$

(R4) consecutive trips do not overlap in time:

$$T_{j,n_j+1} + p_{i_{j,n_j+1}} \leq T_{j+1,0} \qquad \text{for all } j \in \{1, 2, \ldots, m-1\}.$$

In particular, the processing time $p_{i_{j,n_j+1}}$ in (R4) for $i_{j,n_j+1} \in S \cup \{o^2\}$ can be used to model a necessary time period needed to clear and load a second-echelon vehicle.

3.3.3 Flow conservation and satellite capacity

In this section, we show how to check whether a given set of feasible first-echelon and second-echelon routes, including the quantities dropped off and collected by them over time, respect the forward and reverse flows and whether the quantities stored at a satellite exceed the satellite capacity. This shows how the feasibility conditions (F3)–(F5) can be tested.

For that purpose, we denote by Ω^1 the set of all feasible first-echelon trips and by Ω^2 the set of all feasible second-echelon trips. We introduce additional attributes for a first-echelon trip $h = (P, T) \in \Omega^1$:

— the quantity dropped off by the trip h at satellite $s \in S$ at time $t \in \mathbb{T}$:

$$\gamma_{h,s,t}^{\text{fw}} = \begin{cases} L_k^{\text{fw}}, & \text{there exists an index } k \in \{1, 2, \ldots, n\} \text{ with } i_k = s \text{ and } T_k = t \\ 0, & \text{otherwise,} \end{cases}$$

— the quantity that trip h collects at satellite $s \in S$ at time $t \in \mathbb{T}$:

$$\gamma_{h,s,t}^{\text{rv}} = \begin{cases} L_k^{\text{rv}}, & \text{there exists an index } k \in \{1, 2, \ldots, n\} \text{ with } i_k = s \text{ and } T_k = t \\ 0, & \text{otherwise.} \end{cases}$$

Similarly, we define for a second-echelon trip $h = (P, T, L) \in \Omega^2$:

— the quantity that trip h collects from satellite $s \in S$ at time $t \in \mathbb{T}$:

$$\gamma_{h,s,t}^{\text{fw}} = \begin{cases} \sum_{k=1}^n q_{i_k}^{\text{fw}}, & i_0 = s \text{ and } T_0 = t \\ 0, & \text{otherwise,} \end{cases}$$

— the quantity that trip h drops off at satellite $s \in S$ at time $t \in \mathbb{T}$:

$$\gamma_{h,s,t}^{\text{rv}} = \begin{cases} \sum_{k=1}^n q_{i_k}^{\text{rv}}, & i_{n+1} = s \text{ and } T_{n+1} = t \\ 0, & \text{otherwise.} \end{cases}$$

We also define the *load profile* of satellite $s \in S$: Λ_t^s is the quantity processed and stored at satellite s at time $t \in \mathbb{T}$.

Before, we formalize the feasibility conditions (F3)–(F5) (see constraints (3.e)–(3.g) of the MIP model presented in the next section), we explain them with the help of an example.

Example 1 *Figure 3.1 shows a solution for an instance of the 2E-MTVRP-CSRF with two satellites, eight customers, one first-echelon vehicle, and two second-echelon vehicles. Forward and reverse demands of each customer are depicted next to the corresponding vertex. We assume that the customers have non-restricting time windows and that travel times and travel costs coincide (depicted on the arcs). Moreover, the capacity of both satellites is $C_s^{sat} = 10$ and the processing time is $p_s = 1$.*

The first-echelon route performs two trips (in blue), while the two second-echelon routes perform four (in orange) and three trips (in green), respectively. Table 3.1 shows a feasible

time schedule and loading plan for the three routes.
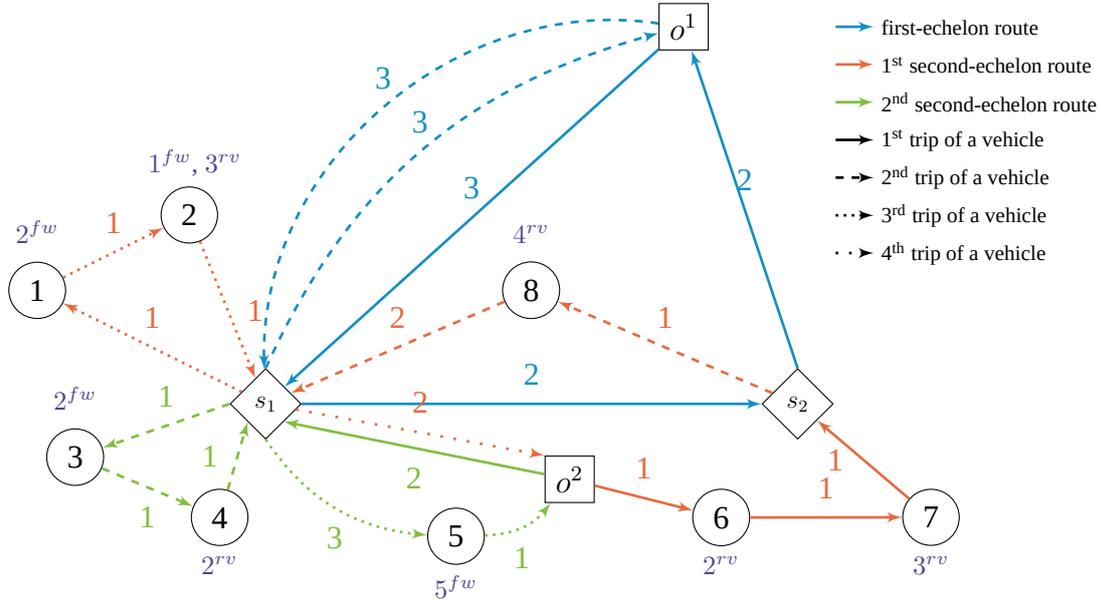
Figure 3.1 – 2E-MTVRP-CSRF solution used in Example 1

First-echelon route (h_1, h_2):

trip index j	1				2		
vertex i_k	o^1	s_1	s_2	o^1	o^1	s_1	o^1
schedule T_k	0	3	6	9	9	14	18
quantity $\gamma^{\text{fw}}_{h_j,i_k,T_k}$	–	10	0	–	–	0	–
$\gamma^{\text{rv}}_{h_j,i_k,T_k}$	–	0	5	–	–	9	–

Second-echelon route (h'_1, h'_2, h'_3):

j	1		2			3			
vertex i_k	o^2	s_1	s_1	3	4	s_1	s_1	5	o^2
schedule T_k	0	2	4	6	7	8	9	12	14
quantity $\gamma^{\text{fw}}_{h'_j,i_k,T_k}$	0	–	2	–	–	–	5	–	–
$\gamma^{\text{rv}}_{h'_j,i_k,T_k}$	–	0	–	–	–	2	–	–	0

Second-echelon route $(h''_1, h''_2, h''_3, h''_4)$:

j	1				2			3				4	
vertex i_k	o^2	6	7	s_2	s_2	8	s_1	s_1	1	2	s_1	s_1	o^2
schedule T_k	0	1	2	3	4	5	9	10	11	12	13	14	17
quantity $\gamma^{\text{fw}}_{h''_j,i_k,T_k}$	0	–	–	–	0	–	–	3	–	–	–	0	–
$\gamma^{\text{rv}}_{h''_j,i_k,T_k}$	–	–	–	5	–	–	4	–	–	–	3	–	0

Table 3.1 – Time schedule and loading plan of the routes in Example 1

Note that there must be at least $p_s = 1$ time units between two consecutive second-level trips. On a FEV, a first-echelon vehicle must stay at least $p_s = 1$ time units at a satellite, however, there is no dedicated time span between consecutive first-echelon trips at o^1. To fulfill the feasibility conditions (F3)–(F5), waiting is mandatory at three visits due to interdependencies: Trip h_2' has to wait for 1 time unit before starting at satellite s_1, because there is no forward demand ready before time 4, i.e., when the dropped-off forward demand of trip h_1 has been processed. The corresponding entries are marked in blue. Similarly, trip h_2^f has to wait for 2 time units at s_1, because it has to collect the reverse demand dropped off by trip h_3^2 at time 13, which is ready for collection at time 14 (red entries). Last, trip h_2'' must wait for 2 time units before unloading at s_1, because there is no free capacity to store its reverse demand. Indeed, the trip has to wait for trip h_3' to free up capacity at s_1 (green entries).

Figure 3.2a shows that these waiting times are necessary to ensure feasibility regarding flows and capacities. The figure visualizes the load profile of satellite s_1. Forward quantities are depicted in red and reverse quantities in blue. Striped areas indicate that the quantities are transferred to the satellite but not processed so that they are not yet ready to be collected. Processed goods are solid. All quantities resulting from first-echelon operations are depicted with arrows on the top of the diagram and all second-echelon operations at the bottom. Regarding the three waiting times, we can see from the figure that:

— *There is no forward quantity ready to collect at the satellite before time 4.*

— *The 9 units of reverse demand collected at time 14 cannot be collected earlier, since the last 3 units dropped-off are not ready at an earlier point in time.*

— *The 4 units of reverse demand dropped off at time 9 cannot be dropped off earlier due to the satellite capacity.*

Conditions (F3)–(F5) are fulfilled, because for each point in time the stored quantity is not greater than the satellites capacity ($C^{sat} = 10$, see Figure 3.2b) and because both the available forward quantities and available reverse quantities are non-negative (see figures 3.2c and 3.2d respectively).

(a) Quantities unloaded, loaded, processed, not yet processed in Example 1

(b) Total load profile in Example 1

(c) Profile of forward goods in Example 1

(d) Profile of reverse goods in Example 1
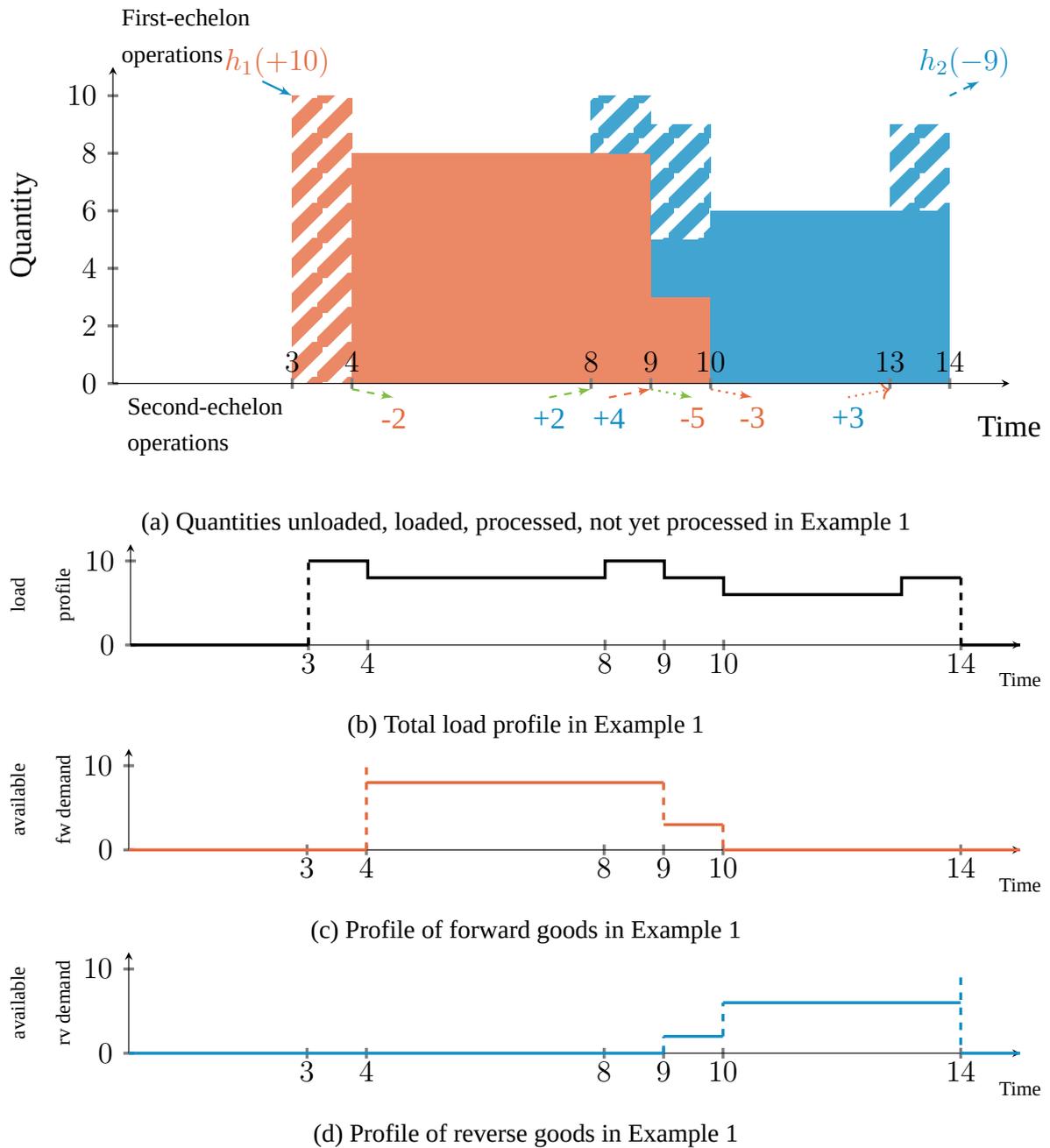
Figure 3.2 – Load profiles of the satellite in Example 1

3.3.4 Trip-based formulation

The *integer programming* (IP) model of the 2E-MTVRP-CSRF presented next serves two purposes: First, it precisely defines the 2E-MTVRP-CSRF. Second, the matheuristic explained

in Section 3.4 uses a restricted and refined version of the IP as one of its components.

The IP can be characterized as a trip-based formulation. Clearly, both sets Ω^1 (feasible first-echelon trips) and Ω^2 (feasible second-echelon trips) are finite but typically extremely large, because trips are defined as combinations of a walk, time schedule, and loading plan (in case of the first-echelon trips). For the IP model, we describe a first-echelon trip $h = (P, T, L) \in \Omega^1$ with the additional attribute $\beta_{h,t} \in \{0, 1\}$, which indicates whether trip h takes place during time $t \in \mathbb{T}$, or not:

$$\beta_{h,t} = \begin{cases} 1, & \text{if } T_0 \leq t \leq T_{n+1} \\ 0, & \text{otherwise.} \end{cases}$$

In a similar manner, a second-echelon trip $h = (P, T)$ has the additional attribute $\alpha_{h,i} \in \{0, 1\}$, which indicates whether trip h visits customer $i \in N$, or not:

$$\alpha_{h,i} = \begin{cases} 1, & \text{there exists an index } k \in \{1, 2, \ldots, n\} \text{ with } i_k = i \\ 0, & \text{otherwise.} \end{cases}$$

Additionally, it is convenient to define two subsets of second-echelon trips for satellites $s \in S$ at times $t \in \mathbb{T}$:

$$\Omega_{s,t}^{2+} = \{h \in \Omega^2 : T_0 = t \text{ and } i_0 = s\} \qquad \text{and} \qquad \Omega_{s,t}^{2-} = \{h \in \Omega^2 : T_{n+1} = t \text{ and } i_{n+1} = s\}$$

$\Omega_{s,t}^{2+}$ ($\Omega_{s,t}^{2-}$) are those second-echelon trips that leave from (arrive at) satellite s at time t.

The IP that we present next comprises two types of variables: the integer variables x_h^1 for $h \in \Omega^1$ count the number of times that trip h is performed (note that two or more first-echelon vehicles can perform an identical trip), and the binary variables x_h^2 indicate whether trip $h \in \Omega^2$ is performed, or not.

Model 3: Integrated two echelons model

$$min \sum_{h \in \Omega^1} c_h^1 x_h^1 + \sum_{h \in \Omega^2} c_h^2 x_h^2$$

$$s.t. \sum_{h \in \Omega^2} \alpha_{h,i} x_h^2 = 1 \qquad\qquad \forall i \in N \qquad (3.a)$$

$$\sum_{h \in \Omega^1} \beta_{h,t} x_h^1 \leq |F^1| \qquad\qquad \forall t \in \mathbb{T} \qquad (3.b)$$

$$\sum_{t \in \mathbb{T}} \sum_{h \in \Omega_{o^2,t}^{2+}} x_h^2 = \sum_{t \in \mathbb{T}} \sum_{h \in \Omega_{o^2,t}^{2-}} x_h^2 = |F^2| \qquad\qquad (3.c)$$

$$\sum_{t' \leq t-p_s} \sum_{h \in \Omega^{2-}_{s,t'}} x^2_h - \sum_{t' \leq t} \sum_{h \in \Omega^{2+}_{s,t'}} x^2_h \geq 0 \qquad \forall s \in S, t \in \mathbb{T} \qquad (3.d)$$

$$\sum_{t' \leq t-p_s} \sum_{h \in \Omega^1} \gamma^{\text{fw}}_{h,s,t'} x^1_h - \sum_{t' \leq t} \sum_{h \in \Omega^{2+}_{s,t'}} \gamma^{\text{fw}}_{h,s,t'} x^2_h \geq 0 \qquad \forall s \in S, t \in \mathbb{T} \qquad (3.e)$$

$$\sum_{t' \geq t+p_s} \sum_{h \in \Omega^1} \gamma^{\text{rv}}_{h,s,t'} x^1_h - \sum_{t' \geq t} \sum_{h \in \Omega^{2-}_{s,t'}} \gamma^{\text{rv}}_{h,s,t'} x^2_h \geq 0 \qquad \forall s \in S, t \in \mathbb{T} \qquad (3.f)$$

$$\sum_{t' \leq t} \sum_{h \in \Omega^1} \gamma^{\text{fw}}_{h,s,t'} x^1_h - \sum_{t' \leq t} \sum_{h \in \Omega^{2+}_{s,t'}} \gamma^{\text{fw}}_{h,s,t'} x^2_h$$
$$+ \sum_{t' > t} \sum_{h \in \Omega^1} \gamma^{\text{rv}}_{h,s,t'} x^1_h - \sum_{t' > t} \sum_{h \in \Omega^{2-}_{s,t'}} \gamma^{\text{rv}}_{h,s,t'} x^2_h \leq C^{sat}_s \qquad \forall s \in S, t \in \mathbb{T} \qquad (3.g)$$

$$x^1_h \in \mathbb{N}_0 \qquad \forall h \in \Omega^1$$

$$x^2_h \in \{0,1\} \qquad \forall h \in \Omega^2$$

The objective minimizes the routing costs of the chosen first-echelon and second-echelon trips. Constraints (3.a) enforce that every customer is served by a second-echelon trip, i.e., condition (F1). Constraints (3.b) and (3.c) are the fleet-size constraints, i.e., condition (F2). Constraints (3.d) guarantee feasible combinations of second-level trips, i.e., they ensure for every satellite and point in time that the number of second-echelon trips that have already left a satellite can never exceed the number of second-echelon trips that have arrived and are ready to leave again. The coupling between the first and second echelon is enforced via constraints (3.e)–(3.g). Forward flow conservation, i.e., condition (F3) is guaranteed by constraints (3.e), and reverse flow conservation, i.e., condition (F4) by constraints (3.f). The satellite capacity constraints (3.g) limit the quantity of goods that can simultaneously be stored at each satellite, see condition (F5). Note that the third and fourth terms consider quantitys that will be collected by first-echelon vehicles in the future and are not yet dropped-off at the satellite by second-echelon vehicles, respectively. The domains of the trip variables are defined by the lasts constraints.

For the flow conservation, i.e., constraints (3.e) and (3.f), it is always feasible to increase the quantities that first-echelon trips drop off and collect at satellites. By doing so, more flexibility is granted to schedule second-echelon trips. However, increasing first-echelon quantities may violate the satellite capacity constraints (3.g). We will exploit these observations in Section 3.4.3 when solving the model with a proper subset of all feasible trips. A restricted and refined model will allow the increase of first-echelon quantities and granting more flexibility to the second echelon, while the resulting surplus at the first echelon does not burden the satellite capacity constraints.

3.4   Solution method

This section presents the matheuristic that we designed to solve the 2E-MTVRP-CSRF. It follows the *fix-and-optimize* paradigm (Helber and Sahling, 2010) with the fundamental idea that the 2E-MTVRP-CSRF can be decomposed in different ways, where parts of the solution are fixed while the remainder is exactly or heuristically optimized.

An overview of the decomposition that we use is provided in Table 3.2. The first subproblem, denoted by *SP1*, optimizes the first-echelon while the second is fixed. Contrarily, the second subproblem, denoted by *SP2*, optimizes the second echelon while the first is partly fixed. We refer to the two LNS algorithms that solve the two subproblems as LNS^1 and LNS^2, respectively. In addition, a MIP solver is used with a refined version of formulation 3 restricted to trip pools generated by the two LNS algorithms.

SP1		SP2		MIP	
Input: solution (x^1, x^2)		**Input:** solution (x^1, x^2)		**Input:** $\bar{\Omega}^1, \bar{\Omega}^2$	
first-echelon trips	[⋆]	first-echelon trips	[+]	first-echelon trips	[⋆, set of]
· walks	[⋆]	· walks	[fixed]	· walks	[fixed]
· schedules	[⋆]	· schedules	[fixed]	· schedules	[fixed]
· loading plans	[⋆]	· loading plans	[⋆]	· loading plans	[fixed]
first-echelon routes	[⋆]	first-echelon routes	[+]	first-echelon routes	[impl. result]
second-echelon trips	[fixed]	second-echelon trips	[⋆]	second-echelon trips	[⋆, set of]
· walks	[fixed]	· walks	[⋆]	· walks	[fixed]
· schedules	[fixed]	· schedules	[⋆]	· schedules	[fixed]
second-echelon routes	[fixed]	second-echelon routes	[⋆]	second-echelon routes	[impl. result]
Solution method:	LNS^1	**Solution method:**	LNS^2	**Solution method:**	MIP solver
Output: solution (x'^1, x'^2) trips H^2		**Output:** solution (x'^1, x'^2) trips H^1		**Output:** solution (x'^1, x'^2)	

⋆: Modifies

⋆: Modifies
+: Partly modifies

⋆: Selects a solution from sets of trips

Table 3.2 – Overview of the decomposition used in the matheuristic (Algorithm 4).

Algorithm 4 shows how the three subproblems and corresponding solution methods interact in the matheuristic that we call *Iterative Two-Stage Heuristic* (ITSH).

ITSH starts with an initial solution $x = (x^1, x^2)$ and empty trip pools $\bar{\Omega}^1$ and $\bar{\Omega}^2$ (Steps (1) and (2)). The current solution is improved by repeating the main loop (Steps (4)–(14)) until the given time limit Γ_{tot} is reached. In each iteration, SP2 is optimized first with LNS^2 (Step (4)), and all generated second-echelon trips are added to $\bar{\Omega}^2$ (Step (5)). Next, SP1 is optimized with LNS^1 (Step (6)) and all generated first-echelons trips are added to $\bar{\Omega}^1$ (Step (7)). When both LNS^1

and LNS^2 fail to improve the best found solution and the MIP execution conditions (defined in Section 3.4.3) are satisfied, the refined version of formulation 3 is solved with trips sets H^1 and H^2 (Step (11)). To keep the size of the MIP reasonable, the trip pools are re-initialized every time a new best solution is found and after each call to the MIP solver (Step (9), described also in Section 3.4.3).

Algorithm 4: Iterative Two-Stage Heuristic (ITSH)

1: $(x^1, x^2) \leftarrow$ initialization
2: $\bar{\Omega}^1 \leftarrow \emptyset, \bar{\Omega}^2 \leftarrow \emptyset$
3: **while** time limit Γ_{tot} not exceeded **do**
4:      $(x^1, x^2, H^2) \leftarrow \text{LNS}^2(x^1, x^2)$
5:      $\bar{\Omega}^2 \leftarrow \bar{\Omega}^2 \cup H^2$
6:      $(x^1, x^2, H^1) \leftarrow \text{LNS}^1(x^1, x^2)$
7:      $\bar{\Omega}^1 \leftarrow \bar{\Omega}^1 \cup H^1$
8:      **if** the best solution has been improved **then**
9:          $(\bar{\Omega}^1, \bar{\Omega}^2) \leftarrow \text{re-initialize}(\bar{\Omega}^1, \bar{\Omega}^2)$
10:     **else if** the MIP execution conditions are satisfied **then**
11:         $(x^1, x^2) \leftarrow \text{MIP}(\bar{\Omega}^1, \bar{\Omega}^2)$
12:         $(\bar{\Omega}^1, \bar{\Omega}^2) \leftarrow \text{re-initialize}(\bar{\Omega}^1, \bar{\Omega}^2)$
13:     **end if**
14: **end while**
15: **return** $(x^1, x^2) \leftarrow$ the best found feasible solution

---

Details follow in the remainder of this longer section, which is structured as follows: Section 3.4.1 elaborates on the two LNS algorithms. As the feasibility check used in LNS is of high importance for the efficiency of the overall matheuristic, we present its details separately in Section 3.4.2. The MIP component, its parameters, execution conditions, how it communicates with the LNS, and the update of the trip pools is described in Section 3.4.3. For the sake of brevity, details about LNS are presented in Appendix B.1, the computation of an initial solution is detailed in Appendix B.1.5, and the refined MIP Model can be found in Appendix B.2.

3.4.1 LNS algorithms

In the LNS metaheuristic, first proposed by Shaw (1998) in a constraint programming context, the current solution is iteratively destroyed and subsequently repaired until a stopping criterion is reached. LNS algorithms have been shown to be successful in optimizing routes and trips, in particular for VRP variants that only comprise standard resource constraints such as route length, capacity, and time-window constraints (Pisinger and Ropke, 2019). Even more

complicated inter-route constraints (an overview is provided in Irnich et al., 2014, Section 1.3.5) can be integrated easily as long as the removal of a customer is always feasible and an efficient feasibility test for the insertion of customers is available (see, e.g., Grangier et al., 2016).

Our LNS algorithms switches between large destruction steps and small destruction steps. Small destruction steps are performed in most iterations to locally and quickly improve solutions (Christiaens and Vanden Berghe, 2020). If the best solution has not been improved for several iterations, a large destruction step is performed (diversification). This small-and-large strategy has been proved highly successful on variants of the generalized vehicle routing problem with time windows (Dumez et al., 2021c). A short synopsis of the types of operators used in our LNS algorithms is provided in Table 3.3, where for each type of operator we indicate whether it is used in LNS^1 and LNS^2 and who first introduced it. Additional information on the LNS algorithms is provided in Appendix B.1 including pseudo code, details of the destroy and repair operators, and the acceptance criterion.

Type		Operator	Algorithm		Source
			LNS^1	LNS^2	
destroy operators	small	split string removal	✓	✓	Christiaens and Vanden Berghe (2020)
		satellite removal	✓	✓	*this paper*
		distance related removal		✓	Ropke and Pisinger (2006b)
		visit removal		✓	*this paper*
		cluster removal		✓	Pisinger and Ropke (2007)
	large	random customer removal		✓	Ropke and Pisinger (2006b)
		random visit removal	✓		Ropke and Pisinger (2006b)
		worst visit removal	✓		Ropke and Pisinger (2006a)
		historical knowledge node removal		✓	Demir et al. (2012)
		trip and route removal	✓	✓	Nagata and Bräysy (2009)
repair operators		random order best insertion	✓	✓	Christiaens and Vanden Berghe (2020)
		largest first best insertion	✓	✓	Christiaens and Vanden Berghe (2020)
		farthest first best insertion		✓	Christiaens and Vanden Berghe (2020)
		closest first best insertion		✓	Christiaens and Vanden Berghe (2020)
		earliest first best insertion	✓	✓	Christiaens and Vanden Berghe (2020)
		latest first best insertion	✓	✓	Christiaens and Vanden Berghe (2020)
		narrow first best insertion	✓	✓	Christiaens and Vanden Berghe (2020)

Table 3.3 – Destroy and Repair Operators

Solving subproblem SP1

The first-echelon subproblem SP1 consists in defining routes for the first-echelon vehicles such that each fixed second-echelon trip can be supplied with its forward demands, and such that its reverse demand is transported to the UDC. To model these demands, we introduce the concept of *First-Echelon Demands (FEDs)*. More precisely, let $h = (P, T)$ be a second-echelon trip starting at time t from a satellite s_1 and ending at time t' at a satellite s_2. For each customer i in this trip with a forward demand $q_i^{\text{fw}} > 0$, we define a *forward FED* $\tau_i^{\text{fw}} = (q_i^{\text{fw}}, t - p_{s_1}, s_1)$. It indicates that the quantity q_i^{fw} needs to be dropped off by a first-echelon vehicle at satellite s_1 not later than time $t - p_{s_1}$ (called the FED due date). Similarly, for each customer i in the trip with a pick-up demand $q_i^{\text{rv}} > 0$, we define a *reverse FED* $\tau_i^{\text{rv}} = (q_i^{\text{rv}}, t' + p_{s_2}, s_2)$. It indicates that the quantity q_i^{rv} needs to be collected by a first-echelon vehicle at satellite s_2 not earlier than time $t' + p_{s_2}$ (called the FED release date). Now, the LNS assign FEDs to FEVs to represent the demand loaded and unloaded in the FEV, i.e., the assigned FEDs define the load plan. Note that this modeling approach implies that the demand of one customer cannot be split over several FEVs in SP1 and SP2.

Example 2 *(cont'ed from Example 1) Trip h_3'' starts at satellite s_1 at time 10, serves customers 1 with demands $q_1^{fw} = 2$ and $q_1^{rv} = 0$ and customer 2 with demands $q_2^{fw} = 1$ and $q_2^{rv} = 3$, and ends at time 13 at the same satellite. In this case, we create three FEDs for this trip: $\tau_1^{fw} = (2, 9, s_1)$, $\tau_2^{fw} = (1, 9, s_1)$, and $\tau_2^{rv} = (3, 14, s_1)$ (recall that the processing time is $p_{s_1} = 1$, which explains the time offset). The FEDs τ_1^{fw} and τ_2^{fw} are assigned to the FEV $(s_1, 3)$ in trip h_1. The FED τ_2^{rv} is served in the FEV $(s_1, 14)$ of trip h_2 .*

Accordingly, SP1 consists of defining FEVs at satellites, the multi-trip routes of the first-echelon vehicles that serve these visits and the assignments of FEDs to FEVs. In this problem, the first-echelon vehicles capacities as well as the satellites capacities, the FEDs due dates, and release dates must be respected. The objective is to minimize the sum of traveling costs. Hence, SP1 belong to the class of resource constrained routing and scheduling problems (Paraskevopoulos et al., 2017) with time-windows, multiple trips, and mixed backhauls.

LNS[1]   To solve SP1, the destroy operators in LNS[1] remove FEDs from the solution. Afterwards, FEVs without any assigned FED are also removed. The repair operators assign unserved FEDs to FEVs, possibly creating additional FEVs in first-echelon routes. We consider three types of insertions for a FED τ at satellite s:

— *Insertion in an existing visit* assigns τ to an existing FEV at satellite s.

— *Insertion in a new visit in an existing trip* creates a new FEV to the satellite s in an existing trip and assigns τ to this FEV.

— *Insertion in a new trip* creates a new trip starting and ending at the UDC and performing a FEV at satellite s in-between. The FED τ is assigned to this FEV.

For each insertion type, all possible insertion positions are evaluated in LNS[1] according to the evaluation process described in Section 3.4.2.

Solving subproblem SP2

The second-echelon subproblem optimizes the routing of the second-echelon vehicles given a (partly) fixed first-echelon. More precisely, we presume the FEVs are fixed, but the satellite and the release (due) date of an FED can be changed. Consequently, it is possible to change the assignment of FEDs to FEVs.

Accordingly, SP2 consists of designing the second-echelon trips and routes, as well as the assignment of FEDs to (fixed) FEVs. The second-echelon routes must respect the second-echelon vehicles capacities and the customers' time-windows, while minimizing the sum of travelling costs. The FED assignments must respect the first-echelon vehicles capacities. Finally, the resulting schedule must respect the satellites capacities and the precedence constraints related to transfers. Hence, SP2 belong to the class of Resource Constrained Routing and Scheduling Problems (Paraskevopoulos et al., 2017) with time windows, multiple trips, multiple depots and mixed backhauls.

LNS[2]   In all LNS[2] destroy operators, when a customer is removed from a second-echelon route, its FEDs are also removed from their FEVs. Conversely, in repair operators, an insertion consists in inserting a customer i in a second-echelon route, update the forward FED τ_i^{fw} and the reverse FED τ_i^{rv} accordingly (date and satellite), assign τ_i^{fw} to an appropriate FEV at the origin satellite of the trip, and assign τ_i^{rv} to an appropriate FEV at the destination satellite of the trip.

LNS[2] employs three types of insertions for a customer i into a second-echelon route:

— *Insertion in an existing trip:* inserts the customer vertex i between two vertices of an existing trip.

— *Insertion in a new trip:* creates a new trip in a route, inserting its origin satellite, customer i and its destination satellite.

— *Split-trip insertions (Grangier et al., 2016):* splits an existing trip by simultaneously inserting customer i into an existing trip and adding a visit to a satellite s between any two

consecutive vertices in this trip. Hence, for each customer insertion position, this opera-
tors seeks the best splitting position. The forward FEDs associated with customers served
before s as well as the reverse FEDs associated with customers served after s needs to be
changed and possibly reassigned to FEVs.

For the three insertion types, all possible insertion positions are evaluated according the
feasibility check process explained in Section 3.4.2. This feasibility process also determines
how FEDs are assigned to FEVs.

3.4.2 Insertion evaluation

In both LNS, a repair operator considers all insertion positions for a given FED/customer.
Consequently, LNS relies on a large number of insertions and thus, an efficient feasibility eval-
uation is required. All insertions are first evaluated in terms of cost and a simple necessary
feasibility condition is checked. Afterwards, all remaining insertion possibilities are ranked in
non-decreasing order of their cost before the more elaborate feasibility tests are performed. Con-
sidering the complexity of the integrated scheduling and routing problem, the feasibility evalu-
ation procedure is based on heuristic tests. This type of approach was also used in Masson et al.
(2014) to improve the performance of an exact feasibility test in the dial-a-ride problem with
transfers. On the VRP with cross-docking and resource constraints, Grangier et al. (2019) show
that a heuristic feasibility test can offer a better compromise than a complete test using constraint
programming, allowing for more iterations of the matheuristic in the same time.

In LNS^2, customers are inserted into routes, but, in addition, the generated FED must be
assigned to FEV, and the assignment of other FED may be changed. To better apprehend the
algorithms used to test insertions, we present a representation of the solutions as a precedence
graph with resource profiles in Section 3.4.2. This representation of the solution is incrementally
modified to evaluate the insertions, and is also modified accordingly with the modification of
the considered solution.

In this section, we illustrate the feasibility process with the following example:

Example 3 *Figure 3.3 represents an incomplete solution for a simple problem with one depot
o, one satellite s and 5 customers.*

*We consider one first-echelon vehicle k_1 of capacity 4 and two second-echelon vehicles (k_2
and k_3) of capacity 2. The satellite has capacity 3, it is the depot for the second-echelon vehicles.
Customers $1^-, 2^-, 3^-, 5^-$ have a forward demand of one unit and customer 4^+ has a pick-up
demand of one unit. For the sake of simplicity, we assume that customers have no time window,*
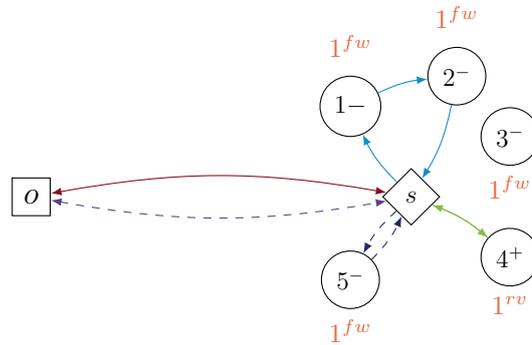
Figure 3.3 – Example for the insertion of customer 3^- in and incomplete solution

that the travel time between each pair of locations as well as the satellite service duration are equal to one time unit, and that customers' service duration is zero. The length of the planning horizon $\bar{t}$ is 9 time units.

In the incomplete solution of Figure 3.3, k_1 performs two back-and-forth trips between the depot and the satellite. It drops-off the demands of customers 1^-, 2^-, and 5^- during its first visit at the satellite. Vehicle k_2 serves customers 1^- and 2^- during its first trip and customer 5^- during its second trip. Vehicle k_3 serves customers 4^- during its only trip. This customer's reverse demand is collected at s by the second visit of k_1. Customer 3^- is not served.

Table 3.4 details the time schedule of the solution presented in Figure 3.3. Each column corresponds to one unit of time in the planning horizon specified by the first line. The next three lines describe the vehicles routes and schedules. The last line indicates the load profile of satellite s.

Time	0	1	2	3	4	5	6	7	8	9
Vehicle k_1	o	s	o			o	s	o		
Vehicle k_2			s	1^-	2^-	s	s	5^-	s	
Vehicle k_3		s	4^+	s						
Satellite load	0	3	1	2	-	-	0	-	-	-

Table 3.4 – Time schedule of the solution of Figure 3.3

This section details the mechanism of the evaluation of customer insertion in an incomplete solution. Section 3.4.2 presents a representation of the solutions with a precedence graph. In Section 3.4.2, we propose a necessary condition and two sufficient conditions for the feasibility of an insertion.

Precedence graph and load profile

Let us consider an incomplete solution x, with a set of routes Θ_x and a set of trips $\Omega_x = \Omega_x^1 \cup \Omega_x^2$ for the first and second echelon, respectively. The precedence graph $G_P^x = (V_P^x, A_P^x)$ associated with x models all precedence relationships between transportation activities and logistic operations that take place at satellites. In addition, let us define the resources profiles Λ_t^s and Ξ_t^r monitoring for each time $t \in \mathbb{T}$ the quantity of goods stored into each satellite $s \in S$ and in each vehicle $r \in \Theta$, respectively. The load profile of the satellite was introduced in Figure 3.2.

The vertex set V_P^x contains: a vertex O representing the start of the time horizon, a vertex O' representing the end of the time horizon, vertices $\{i \in P_h | h \in \Omega_x\}$ representing each vertex in the walks performed by both echelons vehicles, vertices $\{\tau_i^{fw} | i \in N : q_i^{\text{fw}} > 0\}$ modeling forward FEDs, and vertices $\{\tau_i^{rv} | i \in N : q_i^{\text{rv}} > 0\}$ modeling reverse FEDs.

Let us denote by $\mathcal{T}_v$, the time, to be determined, of the beginning of the operation associated with each vertex $v \in V_P^x$. In addition, a time window $[a_v, b_v]$ is associated with each vertex $v \in V_P^x$. It is equal to the time window of the location where the considered activity takes place. Besides, we define $[a_O, b_O] = [0, 0]$ and $[a_{O'}, b_{O'}] = [\bar{t}, \bar{t}]$. The time $\mathcal{T}_\tau$ at which a FED is served is equal to the service time of the FEV to which it is assigned.

The arcs (v, v') of the set A_P^x model precedence constraints between $\mathcal{T}_v$ and $\mathcal{T}_{v'}$ with a weight $W_{v,v'}$. This set consists of the following arcs:

— $\{(O, v) | v \in V_P^x\}$: the arcs from vertex O to any vertex $v \in V_P^x$, each associated with a weight $W_{O,v} = 0$.

— $\{(v, O') | v \in V_P^x\}$: the arcs from any vertex $v \in V_P^x$ to O', each associated with a weight $W_{v,O'} = 0$.

— $\{(i_l, i_{l+1}) | i_l, i_{l+1} \in P_h, h \in \Omega_x^1\}$: the arcs associated with every arc of each first-echelon trip $h \in \Omega_x^1$. An arc (i_l, i_{l+1}) is associated with a weight $W_{i_l,i_{l+1}} = t_{i_l,i_{l+1}}^1 + p_{i_l}$.

— $\{(i_l, i_{l+1}) | i_l, i_{l+1} \in P_h, h \in \Omega_x^2\}$: the arcs associated with every arc of each second-echelon trip $h \in \Omega_x^2$. An arc (i_l, i_{l+1}) is associated with a weight $W_{i_l,i_{l+1}} = t_{i_l,i_{l+1}}^2$.

— $\{(i_{n+1}^{h_l}, i_0^{h_{l+1}})|(h_l, h_{l+1}) \in r, r \in \Theta_x\}$: the arcs between any successive trip of a route, associated with a weight $W_{i_{n+1}^{h_l}, i_0^{h_{l+1}}} = p_{i_{n+1}^{h_l}}$.

— $\{(\tau_i^{fw}, i_0^h)|i \in N : q_i^{\text{fw}} > 0 \land i \in P_h\}$: the arcs associated with each forward demand, from the drop-off at the satellite of the corresponding FED to the departure of the second-echelon vehicle serving the customer, with a weight $W_{\tau_i^{fw}, i_0^h} = p_{i_0^h}$.

— $\{(i_{n+1}^h, \tau_i^{rv})|i \in N : q_i^{\text{rv}} > 0 \land i \in P_h\}$: the arcs associated with each reverse demand, from the arrival at the satellite of the second-echelon trip serving the corresponding customer, to the vertex representing the collection of this FED by a first-echelon vehicle, with a weight $W_{i_{n+1}^h, \tau_i^{rv}} = p_{i_{n+1}^h}$.

Finally, the operations modeled by the vertices $v \in V_P^x$ impact the resources profiles Λ and Ξ at the time $\mathcal{T}_v$ as follows:

— A vertex $D_{r,h}$ modeling the departure of the second-echelon vehicle r for the trip h from the satellite s decreases Λ^s by $\sum_{i \in h} q_i^{\text{fw}}$ and increases Ξ^r by the same quantity.

— A vertex $E_{r,h}$ modeling the arrival of the second-echelon vehicle r at the satellite s after the trip h increases Λ^s by $\sum_{i \in h} q_i^{\text{rv}}$ and increases Ξ^r by the same quantity.

— A vertex i modeling the service at customer i by the second-echelon vehicles r increases Ξ^r by q_i^{rv} and decreases it by q_i^{fw}.

— A vertex $D_{r,h}$ modeling the departure from the depot of the first-echelon vehicle r for the trip h set Ξ^r to $\sum_{i \in h} L_i^{\text{fw}}$.

— A vertex $E_{r,h}$ modeling the arrival of the first-echelon vehicle r at the depot after the trip h reset Ξ^r to 0.

— A vertex V_i modeling the ith FEV to a satellite during the trip h of the first-echelon route r increases Ξ^r by L_i^{rv} and decreases Ξ^r by L_i^{fw}.

— A vertex τ_i^{fw} modeling a forward FED at the satellite s increases Λ^s by q_i^{fw}.

— A vertex τ_i^{rv} modeling a reverse FED at the satellite s decreases Λ^s by q_i^{rv}.

Table 3.5 summarize the vertices of the precedence graph G_P^x, with their time windows and their impact on the resources profiles Λ_t^s and Ξ_t^r.

Name	Set	Time Windows	Impact on ressources
$D_{r,h}$	$\{i_0^h \mid h \in \Omega_x^2\}$	$[0, \bar{t}]$	decreases Λ^s and increases Ξ^r by $\sum_{i \in h} q_i^{\mathrm{fw}}$
$E_{r,h}$	$\{i_{l+1}^h \mid h \in \Omega_x^2\}$	$[0, \bar{t}]$	increases Λ^s and increases Ξ^r by $\sum_{i \in h} q_i^{\mathrm{rv}}$
i	$\{i \in h \backslash \{i_0^h, i_{l+1}^h\} \mid h \in \Omega_x^2\}$	$[a_i, b_i]$	increases Ξ^r by q_i^{rv} and decreases it by q_i^{fw}
$D_{r,h}$	$\{i_0^h \mid h \in \Omega_x^1\}$	$[0, \bar{t}]$	sets Ξ^r to $\sum_{i \in h} L_i^{\mathrm{fw}}$
$E_{r,h}$	$\{i_{l+1}^h \mid h \in \Omega_x^1\}$	$[0, \bar{t}]$	resets Ξ^r to 0
V_i	$\{i \in h \backslash \{i_0^h, i_{l+1}^h\} \mid h \in \Omega_x^1\}$	$[0, \bar{t}]$	increases Ξ^r by L_i^{rv} and decreases Ξ^r by L_i^{fw}
τ_i^{fw}	$\{q_i^{\mathrm{fw}} > 0 \mid i \in h : h \in \Omega_x^2\}$	$[0, \bar{t}]$	increases Λ^s by q_i^{fw}
τ_i^{rv}	$\{q_i^{\mathrm{rv}} > 0 \mid i \in h : h \in \Omega_x^2\}$	$[0, \bar{t}]$	decreases Λ^s by q_i^{rv}

Table 3.5 – Vertices of the precedence graph

The following proposition establishes the relationship between a solution x and its associated precedence graph G_P^x.

Proposition 1 *The solution x, without split at the first echelon, is feasible with respect to constraints F1-F5, Tr1-Tr8, R1-R4 if and only if, for every vertex $v \in V_P^x$, one can find a time $\mathcal{T}_v$ such that:*

— *for each arc $(v, v') \in A_P^x$, $\mathcal{T}_v + W_{v,v'} \leq \mathcal{T}_{v'}$,*

— *the time windows are respected: $v \in V_P^x : \mathcal{T}_v \in [a_v, b_v]$,*

— *the load profile of every first-echelon vehicle is compatible with its capacity: $\forall t \in \mathbb{T}, \forall r \in \theta_x^1 : \Xi_t^r \leq Q^1$,*

— *the load profile of every second-echelon vehicle is compatible with its capacity: $\forall t \in \mathbb{T}, \forall r \in \theta_x^2 : \Xi_t^r \leq Q^2$,*

— *the load profile of every satellite is compatible with its capacity: $\forall t \in \mathbb{T}, \forall s \in S : \Lambda_t^s \leq C_s^{sat}$.*

Example 4 *Figure 3.7 depicts the precedence graph of the solution presented in Figure 3.3.*

For the sake of clarity, we did not print the vertices O and O'. Vertices 1 to 5 represents the customers, along with their demand type denoted by "+" or "-". Vertices of the form $D_{i,j}$ represents the departure of the jth trip of the ith vehicle. Vertices of the form $E_{i,j}$ represent the arrival of the jth trip of the ith vehicle. Finally, V_1 and V_2 are visits to the satellite by the first-echelon vehicle k_1, τ_i^{fw}, with $i \in \{1, 2, 3, 5\}$ represents the drop-off of forward FEDs at the satellite by k_1 and τ_4^{rv} represents the collection of the reverse FED of customer 4 by k_1. The arcs
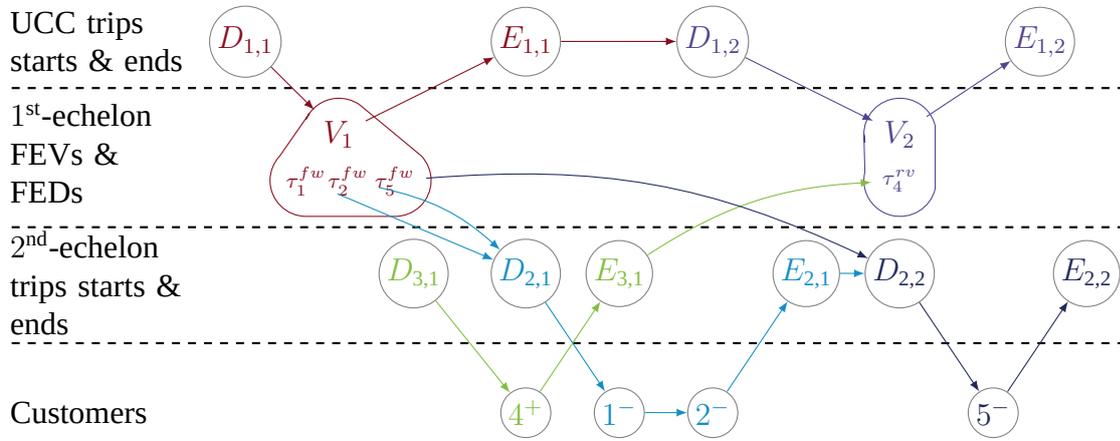
Figure 3.4 – Precedence graph of the solution of Figure 3.3

represent the routes and the precedence relations between the operations at the satellite. The colors identify the trips, as in Figure 3.3.

We define the *precedence graph with fixed assignments* G_{PF}^x of the solution x as its precedence graph G_P^x with additional arcs preventing modification of the order of events taking place at each satellite. That is to say each FED is assigned to its FEV, and arcs with a weight 0 are added between consecutive vertices taking place at the same location ($\{(k,k')|T_k < T_{k'} \wedge i_k = i_{k'} \in S : k \in h, k' \in h' : h, h' \in \Omega_x\}$).

Insertions tests

Sequence of necessary and sufficient conditions   Figure 3.5 summarizes how the necessary condition and the sufficient conditions are called. For a given FED/customer, the condition *NC1* checks the capacity of the vehicle performing the considered trip and a necessary condition for the timing feasibility without synchronization. All insertions that do not satisfy NC1 are rejected. The remaining insertions are sorted in non-decreasing cost order. Their feasibility is tested in this order and the procedure stops as soon as one insertion is proven feasible. The considered insertion is first tested with *SC1*. This condition tests whether the insertion can be performed without modifing the assignment of FEDs to FEVs. If *SC1* is unsuccessful, the considered insertion is tested with *SC2*. This heuristic condition evaluates the same insertion by rescheduling routes and changing the FEDs assignment. The insertion passes if either SC1 or SC2 are positive, otherwise, it is rejected.

Some vertices of the precedence graph may be fixed in time, while others can be rescheduled: In LNS[1] the fixed vertices are the start of the time horizon, the end of the time horizon, the

departure and arrivals of second-echelon trips, and the customer service vertices. In LNS[2] the fixed vertices are the start of the time horizon, the end of the time horizon, and the FEV vertices.

feasible

insertions → NC1 → sort → SC1 → SC2

rejected
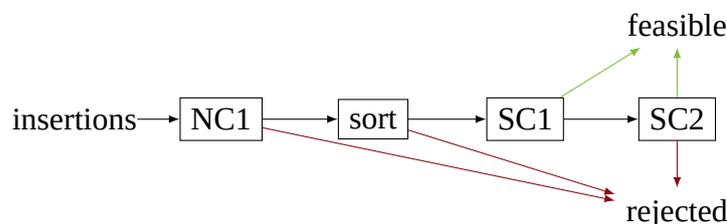
Figure 3.5 – Sequence of necessary and sufficient conditions in LNS feasibility tests.

Necessary feasibility condition   The necessary condition, denoted *NC1*, is based on forward time slacks (Savelsbergh, 1992). A pre-processing step calculates the earliest service time and the latest service time of each vertex of each route, *each route being considered independently of the other routes*. In LNS[1], the time window of FEVs are defined as the intersection of their assigned FEDs' time windows. This time schedule depends solely on the time windows, the travelling times and service durations on the route as well as the start and end of the time horizon. For a vertex, *route-EST* denotes the earliest service time and *route-LST* denotes the latest service time. If the considered insertion violates the route-LST when starting from the route-EST, then the insertion is rejected. In addition, NC1 checks the vehicle capacity. Thus, NC1 can be evaluated in constant time, after a pre-processing in $O(|A_P^x|)$.

Insertion without re-assignment   The first sufficient condition, *SC1*, is composed of two parts: a constant time filter and a heuristic test. SC1 evaluates whether the considered insertion can be performed without changing the assignment of FEDs to FEVs and without changing the order of the operations taking place at each satellite.

The filter is a generalization of the forward time slack in the precedence graph with fixed assignments G_{FP}^x used in SC1. This method is formalized in Masson et al. (2013) for the pickup-and-delivery problem with transfers and extended to the 2E-VRP in Grangier et al. (2016) to take into account the synchronization between multiple vehicles. The algorithm of Savelsbergh (1992) is extended by propagating the earliest service time of a vertex to all its successors in G_{FP}^x and its latest service time to all its predecessors in G_{FP}^x. The calculated service times are called *solution earliest service time* (*solution-EST*) and *solution latest service time* (*solution-LST*). The filter evaluates whether performing the considered insertion respects the solution-

LST when starting from the solution-EST. If it is not the case, then SC1 fails for the considered insertion.

If the test filter is positive, the insertion is temporarily performed in the precedence graph G_P^x. In SP2, an inserted forward FED is assigned to the latest possible (w.r.t the starting time of the trip) FEV with sufficient capacity, and an inserted backward FED is assigned to the earliest possible (w.r.t the ending time of the trip) FEV with sufficient capacity. If no such FEV could be found or if the assignment does not respect the satellite capacity, SC1 fails. In the successful case, the PERT procedure (Miller, 1963) is applied to the fixed precedence graph G_{FP}^x of the resulting solution to compute new time values $\mathcal{T}_v$. SC1 succeeds if the resulting time schedule respects all time windows. Indeed, the satellite capacity was evaluated in the first step of the condition, and the sequence of operations at satellites are maintained by G_{FP}^x structure in the second step. Hence, the capacity of each satellite is necessarily verified. SC1 has a theoretical complexity of $O(|A_P^x|)$.

Example 5 *Figure 3.6 depicts the solution of Figure 3.3 after inserting customer 3^- in the trip of vehicle k_3. The solution-EST of the departure of k_3 is 1, and the solution-LST of vertex 4^+ is 4. Consequently, a 1 time unit detour to visit customer 3^- does not violate these forward time slack and the filter included in SC1 passes. Customer 3^- has a forward demand of 1, thus its forward FED must be assigned to a FEV. The latest possible FEV is the first visit at time 1, and vehicle k_1 has enough remaining capacity. But this assignment does not respect the satellite capacity, and SC1 fails to find a feasible schedule for this insertion. This is illustrated by Figure 3.7 which presents the time schedule of this solution if the forward FED of customer 3^- is inserted into the first FEV according to SC1 procedure. Table 3.6 details the time schedule and satellite load profile that is obtained by the PERT procedure on this graph. The storage capacity of 3 of satellite s is exceeded at time 1.*
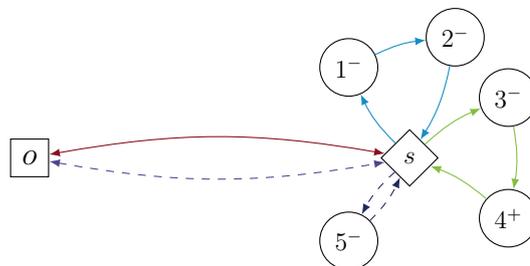

Figure 3.6 – Example: routing of a complete solution.

UCC trips starts & ends

1st-echelon FEVs & FEDs

2nd-echelon trips starts & ends

Customers
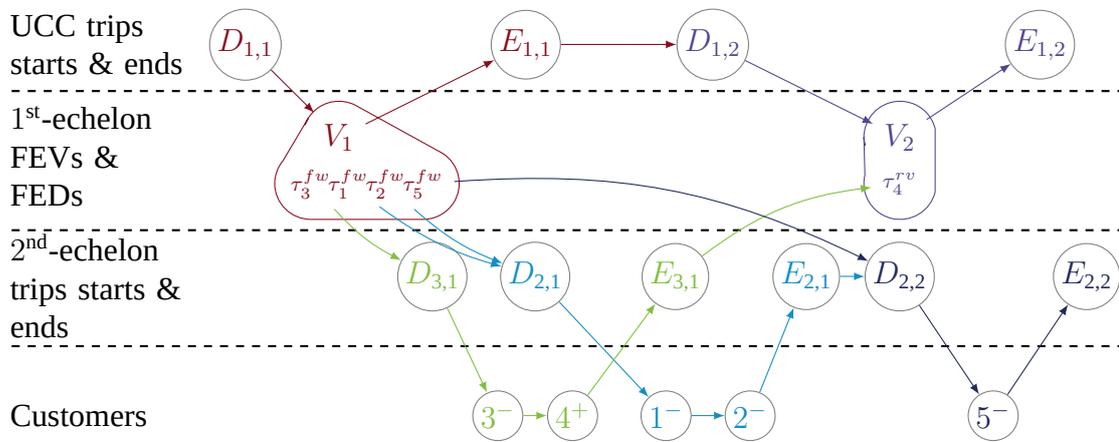
Figure 3.7 – Precedence graph of the solution of Figure 3.6 with the time schedule of Table 3.6.

time	0	1	2	3	4	5	6	7	8	9
Vehicle k_1	o	s	o			o	s	o		
Vehicle k_2			s	1^-	2^-	s	s	5^-	s	
Vehicle k_3			s	3^-	4^+	s				
Satellite load	0	4	1	-	-	2	0	0	-	-

Table 3.6 – Temporary time schedule of the solution of Figure 3.6 after inserting customer 3

Insertion with re-assignments   The second sufficient condition, denoted *SC2*, heuristically performs the considered insertion while allowing modification of the order of operations at satellites and modification of the FEDs assignment to FEVs.

The evaluated insertion is first performed in the precedence graph, like in SC1 but without checking the vehicles and satellite capacities. The inserted vertex is scheduled as early as possible after its predecessor in the route, and this service time is propagated to its successors in the route. This may break some synchronizations between echelons, which need to to be repaired, integrating some possible change in FEDs assignments to FEVs.

To reschedule the solution, SC2 relies on the double-justification (Wiest, 1964). This procedure is a two-phase algorithm shown to be very efficient to improve solutions of resource constraint project scheduling problems (Valls et al., 2005). Starting from an existing schedule, the first phase iteratively schedules all vertices as late as possible by decreasing order of their current service time in the solution. The second phase iteratively re-schedules all vertices as early as possible in non decreasing order of their new service time. For a more detailed and ped-

agogical presentation of the double justification procedure, we refer to (Wiest, 1964). During these processes, all constraints are checked and we do not allow their violations to be increased.

The double justification is applied to try to repair the schedule of every vertex of G_P^x. When rescheduling forward/reverse FEDs at the same FEV, they are considered in an arbitrary order, but linehaul FEDs are postponed first (as late as possible phase) and reverse FEDs are moved as early as possible first. In this process, FEDs can be reassigned to FEVs as follows:

— in the as late as possible scheduling of a forward FED τ_i^{fw}: the FED τ_i^{fw} is assigned to the latest FEV that serves the satellite earlier than the due date of τ_i^{fw} and that is served by a first-echelon vehicle with sufficient remaining capacity.

— in the as late as possible scheduling of a reverse FED τ_i^{rv} : the FEVs of the satellite of τ_i^{rv} are considered in non-decreasing order of service time, starting from the current time of τ_i^{rv}. The capacity of the associated first-echelon vehicles are evaluated. The procedure stops if the minimum remaining capacity of the satellite until the next FEV is lower than q_i^{rv}. Then, τ_i^{rv} is assigned to the latest evaluated FEV with enough capacity.

— in the as early as possible scheduling of a forward FED τ_i^{fw}: the FEVs of the satellite of τ_i^{fw} are considered in non-increasing order of service time, starting from the current time of τ_i^{fw}. The capacity of the associated first-echelon vehicles are evaluated. The procedure stops if the minimum remaining capacity of the satellite until the next FEV is lower than q_i^{fw}. Then, τ_i^{fw} is assigned to the earliest evaluated FEV with enough capacity.

— in the as early as possible scheduling of a reverse FED τ_i^{rv}: the FED τ_i^{rv} is assigned to the earliest FEV that serves its satellite later than the release date of τ_i^{rv} and that is served by a vehicle with sufficient remaining capacity.

SC2 succeeds if the double justification succeeds in finding a schedule that respects the precedence constraints, time windows and capacities of the problem. The theoretical complexity of SC2 is also $O(|A_P^x|)$, but it is considerably slower than SC1 in practice.

Example 6 *Table 3.7 presents the time schedule of Table 3.6 after the first phase (right-alignment) of the double-justification.*

In this as late as possible schedule, the starting time of the second trip of k_2 is postponed from 6 to 7 and the forward FED of customer 5^- is moved from the first visit to the second visit of k_1. In addition, the staring time of the first trip of k_2 is postponed from 2 to 3. The trip of vehicle k_3 cannot be shifted. In this step, the satellite load becomes feasible due to the transfer of the FED of customer 5^- to the second visit of k_1. The precedence graph of this solution is represented by Figure 3.8.

time	0	1	2	3	4	5	6	7	8	9
Vehicle k_1	o	s	o			o	s	o		
Vehicle k_2				s	1^-	2^-	s	s	5^-	s
Vehicle k_3			s	3^-	4^+	s				
Satellite load	0	3	2	0	-	1	1	0	-	-

Table 3.7 – As late as possible time schedule of the solution of Figure 3.6
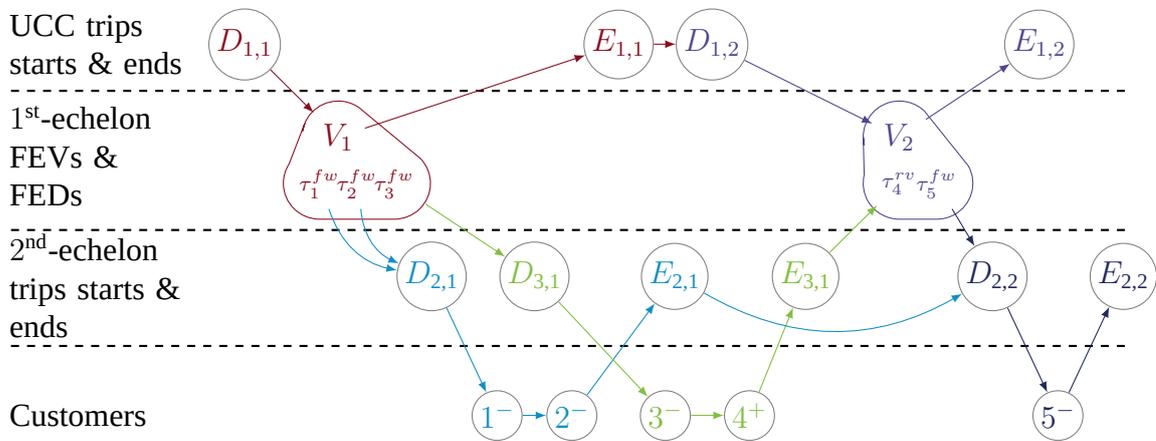
Figure 3.8 – Precedence graph of the solution of Figure 3.6

Table 3.8 presents the time schedule of Table 3.7 after the second phase (left-alignment) of the double-justification. Compared to the as late as possible schedule, the first trip of k_1 starts at 2 instead of 3. Thus, this solution is both feasible and scheduled as early as possible.

time	0	1	2	3	4	5	6	7	8	9
Vehicle k_1	o	s	o			o	s	o		
Vehicle k_2			s	1^-	2^-	s		s	5^-	s
Vehicle k_3			s	3^-	4^+	s				
Satellite load	0	3	0	-	-	1	1	0	-	-

Table 3.8 – Time schedule of the solution of Figure 3.6

3.4.3 Mixed integer program (MIP)

The MIP component of ITSH is called in Algorithm 4, line 11, to assemble trips generated in different iterations of LNS^1 and LNS^2. It is also able to provide a solution where some customer orders are split at the first echelon. The solved MIP is a modified version of Model 3 (see Section 3.3) in which the sets of all possible trips Ω^1 and Ω^2 are replaced by smaller subsets , *the trip pools*, $\bar{\Omega}^1$ and $\bar{\Omega}^2$.

In this section, we present how the MIP interacts with the LNS components: First, we describe the MIP execution conditions that determine when the MIP is solved and more generally the general policy that is used to keep the pools sizes reasonable. Second, we introduce the refined MIP which is solved, allowing some constraints violations penalized in the objective function to allow for infeasible solutions to be explored and improve the solver performance. Third, we introduce how the trip pools are filled and filtered. Finally, we describe how the MIP solution is fed-back into the ITSH framework as a current solution for the next LNS iterations.

MIP execution strategy   A common challenge in the hybridization of a VRP metaheuristic with a MIP component that assemble trips, is the management of the pools of trips. Indeed, if trip pools $\bar{\Omega}^1$ and $\bar{\Omega}^2$ contain too many trips, the MIP solver takes too long to improve the best known solution or spends a lot of time proving optimality. Conversely, if the pools are too small, the solver fails to find an interesting set of trips that can be recombined to form a feasible solution. To handle this difficulty, ITSH uses: a MIP time limit, an adaptive MIP execution conditions, and a pool re-initialization policy.

MIP time limit: The solver is given a limited time budget, denoted by Γ_{MIP}, each time the MIP is solved. Γ_{MIP} depends on the size of the instance (given in Section 3.5.2). Preliminary experiments have highlighted that solving the MIP is much harder when a solution serving all customers has not been found yet. Thus, the value of Γ_{MIP} is increased by 10 seconds for each unserved customer at the second echelon and by 5 seconds for each unserved FED at the first echelon in the current best-known solution. This additional time budget helps the solver, and more generally, ITSH, to find feasible solutions.

MIP execution conditions: The MIP is executed every η ITSH iterations, when the $LNS^2 - LNS^1$ sequence failed to improve the best known feasible solution. Initially, $\eta = 1$ and it is adjusted after each MIP execution depending on the solver result:

— η is increased by 1 if the MIP solution is proven optimal within the time limit, or if the best known feasible solution is improved on two successive MIP executions.

— η is decreased by 1 if the solver fails to solve the root node before the time limit or fails to improve the best-known feasible solution twice in a row. In addition, second-echelon trip copies will not be created during the next call the MIP component.

Pool re-initialization policy: A pool re-initialization consists in clearing the trips pools $\bar{\Omega}^1$ and $\bar{\Omega}^2$, keeping only the trips of the best known solution. In ITSH, this is done: (i) at the end of the $\text{LNS}^2 - \text{LNS}^1$ sequence on one ITSH iteration, if the best known feasible solution has been improved, (ii) every time the MIP is solved.

Restricted and refined MIP   Restricting the trip sets in Model 3 to $\bar{\Omega}^1$ and $\bar{\Omega}^2$ has several consequences: First, the restricted model can be infeasible. To ensure the computation of a partial solution, we introduce auxiliary variables σ_i for not serving customers i. The auxiliary variables are penalized in the refined model's objective function with a high penalty cost M.

Second, in model Model 3, in a first-echelon trip, the quantities dropped off or collected at each FEV of the trip is given. Hence, a first-echelon trip that drop off or collect more than the forward/reverse demand of the second echelon may not be used in a solution if its load planning generates some satellite capacity violation. To enable first-echelon trips to be used in a solution even if their load planning exceeds the actual demand from the second echelon, we introduce the auxiliary variables u, which models the surplus at the first echelon. More precisely, the variable $u^{\text{fw}}_{s,t}$ ($u^{\text{rv}}_{s,t}$) decreases the quantity of goods dropped-off (collected) by first-echelon vehicles visiting the satellite s at time t in the reformulated flow-conservation (3.e) ((3.f)) and satellite-capacity constraints (3.g).

Third, in model 3, fleet size, flow conservation and satellite capacity constraints (3.b), (3.c)–(3.g) have to be defined for all possible points in time $t \in \mathbb{T}$. In the restricted and refined MIP, we reduce the number of these constraints by defining them only for times that correspond to trip ends or trip starts in $\bar{\Omega}^1$ and $\bar{\Omega}^2$.

Fourth, to speed-up the early branching phases, for each satellite $s \in S$, we introduce the binary variable ζ_s to decide whether the satellite s can be used or not. The solver is configured to branch first on these variables. An additional constraint enforces that if ζ_s is set to 0, then no first-echelon trip visiting satellite s, and no second-echelon trips starting or ending at s, can be selected.

All technical details including the complete revised formulation, precise definition of variables, constraints, and index sets are elaborated in Appendix B.2.

Pool accumulation and update   All trips generated in the course of the LNS algorithms are added to the trip pools $\bar{\Omega}^1$ and $\bar{\Omega}^2$ (Line 23 of Algorithm 5 in Appendix B.1). In the course of the LNS, four time schedules are computed for each trip: route-EST, route-LST, solution-EST, and solution-LST. To enable a synchronization with trips generated on various LNS iterations, we create trip variables with identical walks but differences time schedules. For each first-echelon trip and each second-echelon trip in the pool, we generate a trip variable with the solution-EST schedule. For a second-echelon trip in the pool, we create additionally a trip variable with route-EST time schedule and a trip with the route-LST time schedule, each with probability 50 %.

We also modify the load plan of all first-echelon trips to grant more flexibility. More precisely, we build load plans that fulfill $\sum_{k=1}^{n} L_k^{\text{fw}} = \sum_{k=1}^{n} L_k^{\text{rv}} = Q^1$. The missing forward and reverse quantity to reach the vehicle capacity is computed and evenly distributed among all visited satellites such that the vehicle capacity is never exceeded. Note that the load plan of a first-echelon trip always remains feasible if we increase the dropped-off amount at the first visited satellite and the collected amount at the last visited satellite to meet the condition $\sum_{k=1}^{n} L_k^{\text{fw}} = \sum_{k=1}^{n} L_k^{\text{rv}} = Q^1$. However, it is not always possible to evenly distribute the missing loads and remain feasible. Thus, if it is impossible to increase evenly, the remaining quantities are put on the first/last visits.

Communication of a MIP solution to the LNS   Communicating the solution of the refined MIP to both LNS algorithms amounts to perform the following tasks:

(1) *Assemble the chosen trips into routes:* this is done by solving a simple assignment problem: assigning trip starts to trip ends, with an artificial route start vertex and artificial route end vertex.

(2) *Correct the quantities dropped-off and collected at each stop of the chosen first-echelon trips:* the value of the auxiliary u-variables is subtracted from the quantities dropped-off or collected by first-echelon trips at their corresponding FEV. In case a u-variable impacts several FEVs, the surplus quantity is iteratively removed from the FEDs, in the limit of the quantities they drop-off/collect, in arbitrary order.

(3) *Assign FEDs to FEVs:* a generalized assignment problem is solved with the MIP solver to assign the FEDs to the quantities dropped off and collected at each FEV . If it exists no complete feasible assignment, we assign as many FEDs at possible and customers corresponding to unassigned FEDs are removed and placed in the request bank. If this (incomplete) assignment violates satellite capacity constraints, we iteratively remove pairs of FEDs and customers until feasibility is restored. Removed customers are stored in the request bank.

3.5   Computational results

In this section, we report the results of computational experiments. Section 3.5.1 describes the new instances that we have generated. The LNS and MIP components are evaluated in Section 3.5.2. In Section 3.5.3, we compare our algorithm on 2E-VRPTW instances of Marques et al. (2020a). Finally, sensitivity analyses that provide managerial insights are conducted in Section 3.5.4.

The algorithm is coded in C++ and compiled with g++ 5.4.0. We use IBM Ilog CPLEX 20.1.0 (IBM, 2018) as the MIP solver. Options 'branch up first' and 'emphasize finding high quality feasible solutions earlier' of CPLEX are used. The experiments were performed on a PC running Linux, Ubuntu 20.04.2 LTS, equipped with an Intel Xeon Gold 6230 @ 2.10GHz. A single thread is used by our code and CPLEX. On the 2E-VRPTW, distances were round up to two digits, and on the 2E-MTVRP-CSRF, distances were round up to zero digits (i.e. integer).

3.5.1   Instances

As there are no 2E-MTVRP-CSRF instances (with reverse flow, satellite capacities, and multiple first-echelon trips), we have generated new ones with the following properties:

— The instances have a Euclidean representation over a $[0, 100] \times [0, 100]$ grid. Customers are placed randomly with a uniform distribution on the grid, while the UDC is always located at $(50, 100)$. The second-echelon depot and all satellites are randomly placed in $[20, 80] \times [20, 80]$ with a pairwise minimum distance of 20. Times (in minutes) and distances (=routing costs) are computed as rounded up Euclidean distances.

— The planning horizon is $\mathbb{T} = \{0, 1, \ldots, 600\}$. Customer time windows are assigned with equal probability: early $[0, 300]$, late $[300, 600]$, or two-hours during the day as $[60 + 120\psi, 180 + 120\psi]$ with $\psi \in \{0, 1, 2, 3\}$. The customer service time is 5 minutes, and all satellites have a service time of 15 minutes.

— Customers have either only forward demand, only reverse demand, or both, with probability 50%, 25%, and 25%, respectively. This demand is uniformly drawn from $\{1, 2, 3, 4\}$. The first(second)-echelon vehicle capacity is set to 75 (10).

We systematically vary the following parameters to obtain 36 *instance groups*:

— The number of customers is either 50 or 100.

— 50-customer instances have a fleet of 1 (5) first(second)-echelon vehicle(s), and 100-customer instances have a fleet of 2 (10) first(second)-echelon vehicles.

— The number of satellites is 2, 4, or 8.

— For all satellites, the satellite capacity is set to the same value $C^{sat} \in \{20, 25, 35, 50, 70, 500\}$. Since 500 is an upper bound for the total demand, this value represents uncapacitated satellites, i.e., $C^{sat} = \infty$.

This gives $2 \times 3 \times 6 = 36$ groups of instances. We generate ten instances per group resulting in 360 2E-MTVRP-CSRF instances that are available at `https://logistik.bwl.uni-mainz.de/research/benchmarks/`.

3.5.2 Component Evaluation

First, we evaluate different configurations of the MIP component. We limit the test instance set to the 60 instances with a tight satellite capacity of $C^{sat} = 25$ (recall that 25 is one third of the capacity of a first-echelon and 2.5 times the capacity of a second-echelon vehicle). Most of these instances are feasible, but the determination of a feasible solution is non-trivial. The overall time limit is 700(2000) seconds for 50(100)-customer instances and the MIP solver has a time budget $\Gamma_{\mathrm{MIP}} = 60(150)$ seconds in each iteration. We compare the performance of the complete configuration (ITSH) as presented in Section 3.4.3 with the following restricted configurations:

— *noMIP*: the MIP is not used.

— *noAdaptTime*: the time budget of the MIP is not dynamically adapted, it is always set to Γ_{MIP}.

— *noVAR-u*: the additional variables u are not used. Nonetheless, variables σ, used for the penalization of unserved customers, are necessary during early stages.

— *noSplit*: the MIP decomposition allows the split of customers' demand over different FEVs. Here, these split-solutions of the MIP are considered infeasible.

Table 3.9 presents the results, where for each instance, the best solution found out of 5 runs is taken into account. Each line corresponds to a group of instances defined by the number $|N|$ of customers and number $|S|$ of satellites. For each configuration, the entry #f describes the number of instances for which a feasible solution has been computed with the configuration. For each of the aforementioned feasible solution, we compare their cost to the cost of the respective best solutions found by ITSH. While ITSH provides the reference solutions (average costs are presented in column 'cost'), the average percentage gap to the ITSH solution, computed as $100 \cdot (z_{conf} - z_{ITSH})/z_{ITSH}$, is presented in columns 'Gap' for all three alternative configurations.

$|N|$	$|S|$	#inst	ITSH		noMIP		noAdaptTime		noVAR-u		noSplit	
			#f	cost	#f	Gap	#f	Gap	#f	Gap	#f	Gap
50	2	10	10	1 965.0	8	2.01	10	0.65	9	−1.03	10	0.71
	4	10	10	1 725.7	9	2.36	10	0.65	10	2.36	10	0.83
	8	10	10	1 663.1	10	2.54	10	0.35	10	1.24	10	0.09
Total		30	30	1 784.6	27	2.33	30	0.55	29	0.92	30	0.55
100	2	10	9	3 402.1	6	3.23	8	3.66	6	2.54	9	0.14
	4	10	10	3 021.8	9	2.22	9	0.63	10	0.91	10	0.33
	8	10	10	2 572.0	10	3.94	10	−0.21	10	1.66	10	0.03
Total		30	29	2 984.7	25	3.15	27	1.30	26	1.61	29	0.17

Table 3.9 – Comparison of different MIP configurations

The complete configuration ITSH computes feasible solutions for 59 of the 60 instances, which is better than any restricted configuration.

The solutions computed with ITSH are better than the solutions computed with any other configuration. The MIP appears as an essential components of ITSH, more particularly with the adaptive time budget (average gap of approximately 2.7% over all instances). Moreover, the additional variables improve the quality of the solutions compared to *noMIP*. Last, the split of customers' demand seems to have only a small impact on the solution quality. It was to be expected since the largest gain from the split delivery VRP, compared to the VRP, occurs when the customer demands are larger than half of the vehicle capacity (Archetti and Speranza, 2008), while, in our case, customers demand are much smaller than first-echelon vehicle capacity.

Second, we evaluate the impact of the multi-level feasibility checking strategy for the second-echelon LNS presented in Section 3.4.2. Recall that only LNS[1] uses both tests SC1 and SC2 (LNS[1] uses only SC1). Figure 3.9 visualizes how potential insertions are filtered by the insertion feasibility check. We considered an instance with 50 customers, two satellites, and satellite capacity 25. The execution of the ITSH includes more than 17 millions insertions attempts. Most infeasible insertions (>16 millions) are detected by NC1 and directly rejected. The subsequent sort of the insertions further reduces the number of calls to SC1 from 1.42 million to 654k. Then, SC1 identifies approximately half (29k) of the feasible insertions that are performed. Finally, the very time-consuming SC2 identifies the other half (27k).
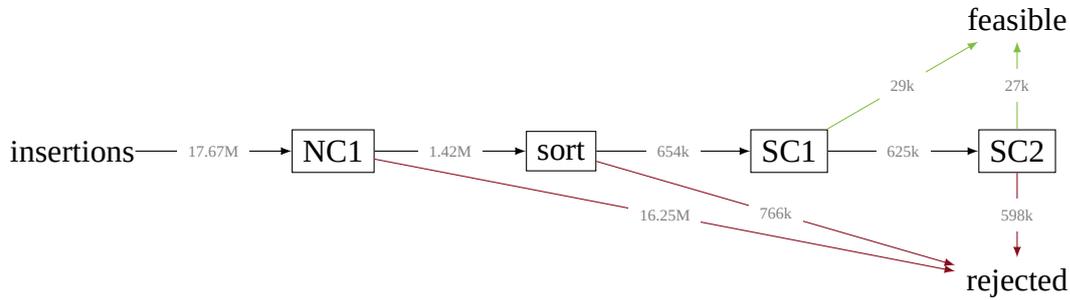
Figure 3.9 – Example of the usage of the feasibility tests

We now analyze the impact of SC1 and SC2 on a restricted test set consisting of the 30 instances with 100 customers and satellite capacity 25:

— When SC1 is deactivated in LNS[2], the average cost of a solution increases by 1.8%. The impact on cost is even more striking given that approximately 90% of the computation time is consumed by the MIP solver and not the LNS[2]. A single evaluation of SC2 takes approximately 450 times more time than an evaluation of SC1. Hence, SC1 is a very powerful speed-up procedure to to fastly detect feasible insertions without re-assignment.

— When SC2 is deactivated in LNS[2], many feasible insertions remain undetected, because all undecided insertions after SC1 are rejected. As a result, not a single feasible solution at all is determined with ITSH. The solution process is stuck producing only new solutions similar to the initial infeasible solution. Recall that SC1 assumes that the order of operations taking place at each satellite cannot change. In particular, without SC2 and only SC1, the order of second-echelon arrivals and departure is fixed relative to the current first-echelon visits, which is highly constrains the insertion of vertices into alternative positions.

3.5.3 Comparison on 2E-MTVRPTW benchmark

We compare configuration ITSH with state-of-the-art algorithms for the 2E-VRPTW. Grangier et al. (2016) modified and extended instances from the well-known benchmark set of Solomon by placing the UDC to the coordinate (50,150) and adding eight additional satellites. Vertex 0 represents the second-echelon depot. Customer time windows are shifted to account for first-echelon and satellite processes. Recently, Marques et al. (2020a) extended this benchmark set by creating smaller instances with 25, 50, and 75 customers. They consider only the first customers in the respective 100-customer instance. The smaller instances have between 4 and 6

satellites.

We can directly compare our method with the results of the *branch-price-and-cut* (BPC) algorithm of Marques et al. (2020a), who minimize the total cost including a fixed cost per vehicle of 50 (25) per first(second)-echelon vehicle. For a fair comparison, we only consider the instances for which Marques et al. report a feasible solution value, because we use their number of vehicles as our fleet-size limit.

Table 3.10 summarizes the comparison with the exact BPC algorithm proposed in Marques et al. (2020a) with a 10-hour time limit.

	$\lvert N \rvert = 25$ $\Gamma_{\text{tot}} = 30$			$\lvert N \rvert = 50$ $\Gamma_{\text{tot}} = 120$			$\lvert N \rvert = 75$ $\Gamma_{\text{tot}} = 500$			$\lvert N \rvert = 100$ $\Gamma_{\text{tot}} = 800$		
Group	#	BPC time	Gap	#	BPC time	Gap	#	BPC time	Gap	#	BPC time	Gap
c100	0/9	23	0.50	0/9	161	0.29	3/8	9 621	1.01	0/7	16 503	1.91
r100	0/12	62	0.15	0/12	882	1.13	2/10	27 049	0.97	0/3	19 970	0.41
rc100	0/8	10	0.55	0/8	89	1.10	0/7	23 168	1.07	3/4	36 092	−0.75
c200	0/8	585	0.39	1/8	8 294	2.41	0/4	7 809	2.35	0/1	15 935	0.44
r200	0/8	5 169	3.34	0/1	1 714	0.07						
rc200	0/8	356	5.16	0/6	7 847	2.12						
All	0/53	942	1.54	1/44	2 907	1.30	5/29	18 932	1.19	3/15	22 382	0.80

Table 3.10 – Performance on the 2E-VRPTW instances compared to the BPC algorithm of Marques et al. (2020a).

The two numbers x/y in columns headed '#' are the number y of instances considered and the number x of new best solutions computed with ITSH (if no feasible solution was provided by the BPC algorithm, the cells are left blank). Our algorithm has a fixed time budget Γ_{tot}(in seconds) controlled by the number $\lvert N \rvert$ of customers (see entries Γ_{tot} in the second line of the table's header; associated CPLEX time limits Γ_{MIP} are set to 2, 10, 40, and 60 seconds, respectively). In contrast, run times of the BPC algorithm are unpredictable, and average times (in seconds) are presented in columns 'BPC time'. Moreover, the table shows the average gaps ('Gaps'; in percent) between the best solutions found by our algorithm (five independent runs per instance) and the best solutions found by Marques et al. (2020a). Detailed instance-by-instance results can be found in Appendix C.1.

ITSH delivers feasible solutions to all instances for which the BPC algorithm also computes feasible solutions. Moreover, ITSH finds nine new best solutions for the benchmark set (leading to a negative gap in the group rc100 with $\lvert N \rvert = 100$). We further interpret the results in the

following way. Instances with more customers, wider time windows and longer routes (series 2) are more difficult for the BPC. As a heuristic and in comparison to the BPC algorithm, ITSH can better provide good feasible solutions so that average gaps are even decreasing for instances with more customers. However, some larger gaps (up to 6.41% for a 25-customer instance in group rc200) show that ITSH can still fail to reach some optimal solution values computed by the BPC (in several hours of computation time).

3.5.4   Sensitivity analyses

The consideration of capacitated satellites and reverse flows in the two-echelon system are in the focus of this work. We now study their impact on costs in sensitivity analyses. All results are based on the best solution computed with the complete ITSH configuration in five independent runs for each instance.

Cost of satellites capacity   Figure 3.10 depicts the impact of satellites capacity. In both sub-figures (for 50- and 100-customer instances), the horizontal axis represents the capacity of the satellites and the vertical axis represents the average increase in cost (in percent) compared to the respective uncapacitated instance with $C^{sat} = \infty$. The figures show three plots with the result for 2, 4, and 8 satellites, respectively. To put these results in perspective, the average total forward demand is of 188 and the average total reverse demand is of 124 on the instances with 100 customers. Detailed results can be found in Section C.2 of the Appendix.

(a) Instances with 50 customers   (b) Instances with 100 customers
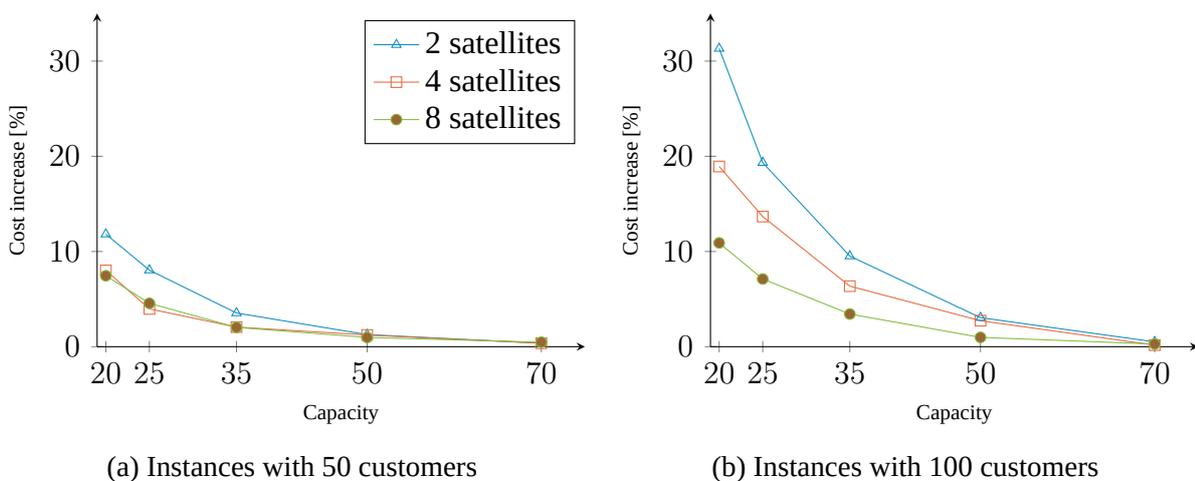
Figure 3.10 – Average cost increase (in percent) compared to the respective instance with no satellite capacity restrictions.

As expected, the average cost increase declines with larger satellite capacities. The fewer satellites are available, the smaller the cost difference (with the exception that 4 and 8 satellites make nearly no difference for 50-customer instances). In comparison, the cost increase is larger for 100-customer than for 50-customer instances, which can be explained by the fact that the capacity of a satellite is constant in contrast to the doubled total demand. Note that we disregarded some instances with two satellites and a capacity of 20 and 25, because no feasible solutions could be found due to the (overly) restrictive capacity.

The convex shape of the curves indicates that the benefit from additional capacity units is strongly decreasing: For the 100-customer instances, the five units more for extending the capacity from 20 to 25 have a much higher impact (8.08%) than the 20-unit increase from capacity 50 to 70 (1.96%). Moreover, the difference between satellite capacity 70 and uncapacitated satellites becomes negligible (0.32%).

Doubling the capacity leads to a much larger cost decrease than doubling the number of satellites. Nevertheless, a small number and size of satellites can be a reasonable design given the outstanding prices of space in some cities (e.g., £$14355/m^2/year$ in London, Furmanik, 2019).

Gain of simultaneously handling forward and reverse flows   To evaluate the gain that results from the integration (i) of forward and reverse flows, we create two distinct instances with only the forward demands and the reverse reverse demands, respectively. We denote by c^i the cost of a best solution found by ITSH for the original instances and by c^{fw}, and c^{rv} the cost found for the new instances with only forward or reverse demands. In Table 3.11, Columns 3 to 8 present the average gain of the integrated approach, computed as $(c^{fw} + c^{rv} - c^i)/c^i$, grouped by the number of customers and satellites as well as satellite capacity. Note that the comparison is biased, since we assumed that both forward flow and reverse flow can use the complete satellites capacity when forward and reverse are managed independently.

On the opposite, Column 9 evaluates the gain from the integration by considering that the integrated problem can use the full satellite capacity (50) while each sub-instance can use only part of the capacity. Since the total forward demand is larger than the total reverse demand, we decided that forward instances use satellites of capacity 35 while reverse instances use satellite of capacity 25.

		Satellite capacity C^{sat}						
$|N|$	$|S|$	20	25	35	50	70	∞	$50 \rightarrow 35 + 25$
50	2	23.55%	27.99%	33.08%	35.74%	36.89%	37.33%	36.11%
	4	30.80%	34.07%	35.85%	36.50%	37.45%	37.83%	37.16%
	8	31.50%	33.12%	35.09%	36.08%	36.21%	36.59%	36.57%
Total		29.55%	31.73%	34.67%	36.11%	36.85%	37.25%	36.61%
100	2	11.27%	22.75%	31.04%	37.55%	39.74%	40.38%	39.96%
	4	22.47%	27.02%	32.98%	36.05%	39.00%	39.13%	38.14%
	8	33.74%	37.16%	39.53%	42.19%	42.95%	43.24%	43.63%
Total		25.30%	29.19%	34.52%	38.60%	40.56%	40.92%	40.58%

Table 3.11 – Average gain resulting from simultaneously handling forward and reverse flows.

The gain from the integration tends to increase with the total space provided by all satellites and the total quantity of goods to carry. It increases from 11% with two small satellites for 100 customers, to 43% with 8 large satellites for 100 customers. On one hand, the gain consistently increases with the satellite capacity. On the other hand, the gain tends to increase with the number of satellites, but this trend is not consistent for the 50-customer instances and $|S| = 4$ or 8. We can only explain this with the fact that the instances with eight satellites are much more difficult for ITSH compared to those with only four satellites. The point is that, for instances with $|S| = 8$, not all satellite have to be used.

Furthermore, gains presented in columns 3 to 8 assume that both forward flow and backward flow can use the satellites entirely when they are managed independently. This would result in clearly unrealistic solutions in practice. Thus, in Column 9, we evaluate the gain from the integration if each type of flow could use only a part of the satellites capacity . The gain from the integration then rises to 43% if 8 satellites are used for 100 customers.

3.5.5   Insights on instances

To better apprehend the previous tables, this section focuses on the time spent by goods at the satellites. We compute the average time between the arrival of a FED at a satellite and the departure of its associated second-echelon vehicle (or the opposite for a backward demand). This time includes the service duration at the satellite (15 units out of a planning horizon of 600

units). Table 3.12 shows the average time spent by goods at the satellites, in instances with 100 customers, with respect to the number of satellites and their capacity. Infeasible instances are not taken into account.

	Satellite capacity C^{sat}					
# satellites	20	25	35	50	70	∞
2	38.5	44.3	58.3	85.0	116.5	135.2
4	53.9	70.6	89.7	101.0	126.1	128.8
8	67.0	82.1	95.0	114.8	117.6	118.1
All	56.1	66.4	81.0	100.3	120.1	127.4

Table 3.12 – Average time spent by the goods at satellites

In general, the waiting time of goods at satellites increases with the satellites capacity or the number of satellites, the impact of the number of satellites being significantly greater with tighter satellites capacities.

Finally, figures 3.11a and 3.11b show the quantity of goods stored at each satellite over time with a satellite capacity of 25 and 500, respectively. In the following, the first instance is called the *tight capacity 2E-MTVRP-CSRF*, while the second is called the *uncapacitated 2E-MTVRP-CSRF*. Each curve corresponds to a satellite, the colors are the same in the two figures. In addition, time-series at the top indicate which type of vehicle operates at each satellite: a square for a first-echelon visit, a triangle for a second-echelon vehicle arrival, and a pentagon for a second-echelon vehicle departure. This comes from the best solutions found for an instance of class B with 100 customers. In this instance, the total forward demand is of 188 and the total backward demand is of 138. Because both solutions use four satellites, the average quantity of goods passing by each satellite is of 81.5.

The cost of the tight-capacity 2E-MTVRP-CSRF solution represented by Figure 3.11a is 3321: 570 for the first echelon and 2751 for the second-echelon. The cost of the uncapacitated 2E-MTVRP-CSRF solution represented by Figure 3.11b is 2694: 474 for the first echelon and 2220 for the second-echelon. The second-echelon fleet has 10 vehicles. They are all used in the constrained case and only 7 vehicles are used in the uncapacitated 2E-MTVRP-CSRF.

Both solutions are using approximately the same number of trips: 3 and 4 first-echelon trips with 10 and 12 first-echelon visits, and 39 and 40 second-echelon trips. But when the satellite capacity is tight, all trips must be synchronized accurately to avoid exceeding these capacities. Indeed, the average time spend by goods at satellite in the tight capacity 2E-MTVRP-CSRF is

of 81, while it is of 44 in the uncapacitated case.

Let us consider how goods are spread over satellites in these solutions: In the tight capacity 2E-MTVRP-CSRF, the quantity of goods processed by each satellite is: 82, 127, 68, 49. In the uncapacitated 2E-MTVRP-CSRF, the quantity of goods processed by each satellite is: 52, 154, 72, 48. With a tight capacity, the goods have to be spread over satellites because the best satellite location is overloaded. The goods are better spread in time to take account of the tight capacity. Indeed, the quantity of goods arriving at satellites during each block of 100 time units in the tight capacity 2E-MTVRP-CSRF solution are: 70, 68, 41, 104, 38, 9. In the uncapacitated case, these values are: 135, 24, 72, 46, 17, 36. The standard deviations of these two sequences are 32 and 43, respectively, showing a significantly higher dispersion in the uncapacitated 2E-MTVRP-CSRF.
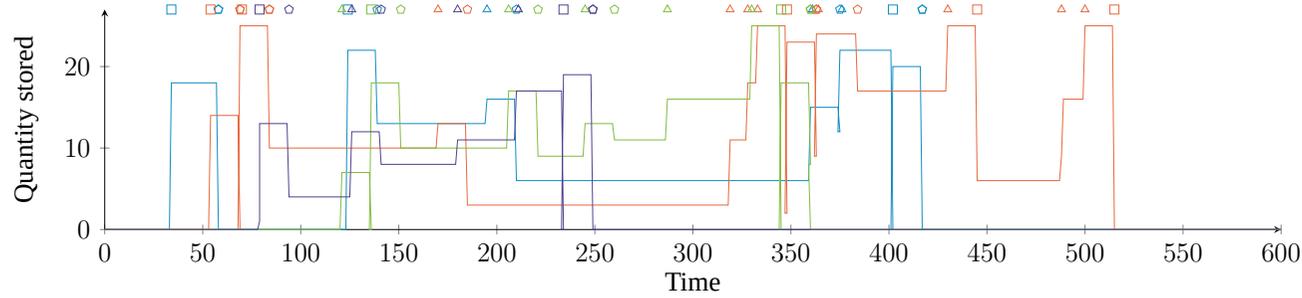
(a) Satellite usage with 25 units capacity
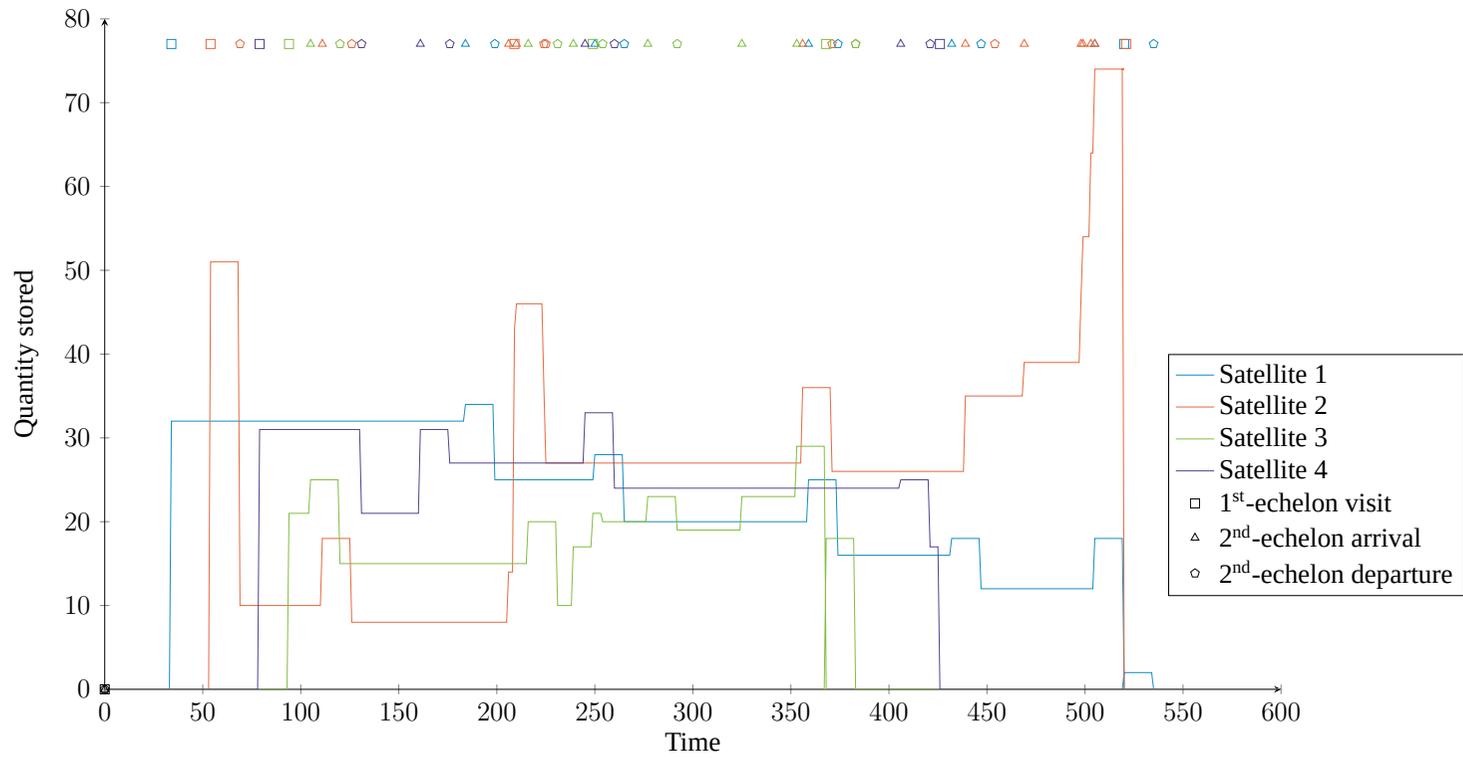
(b) Satellite usage with unlimited capacity

Figure 3.11 – Satellite usage over time

3.6   Conclusion

Two-echelon distribution systems are often introduced as the cornerstone of city logistics. In this paper, we study major functionalities that are generally ignored in most operations research algorithms for this type of system: forward and reverse flows, multiple vehicle trips and storage capacities at satellites. We introduce the 2-Echelon Multi-Trip Vehicle Routing Problem with Capacitated Satellites and Reverse Flows (2E-MTVRP-CSRF), which models how these new features can be integrated in an optimization problem.

To solve this problem, we designed a decomposition-based matheuristic: Iterative Two-Stage Heuristic (ITSH). ITSH uses LNS to improve the routes at both echelons, taken separately, and a solver to solve a MIP model which recombines trips generated for the two echelons on distinct LNS iterations. To efficiently test the feasibility of insertions in a partial solution of the problem, the LNSs rely on incremental implementation of algorithms inspired from the scheduling literature: the PERT algorithm and the double-justification. The MIP is formulated and adapted such that it is able to recombine many trips generated by the LNS algorithms in a reasonable amount of time. Both types of methods have been integrated using adaptive mechanisms so that ITSH adequately shares its solving time between the search for efficient new trips and the combination of existing trips.

To evaluate ITSH on the 2E-MTVRP-CSRF, we generated instances with up to 100 customers and 8 satellites [1]. In our experiments we find that:

— ITSH provides good results on instances of the 2E-MTVRPTW in a comparison to the exact algorithm of Marques (2020).

— The combination of the heuristic and exact components is a key success factor of the algorithm.

— The routing costs are very sensitive to the satellite capacity. On 100 customers instances with two satellites, moving from a large capacity of 70 to a tight capacity of 20 generates a 30% cost increase on average and feasibility issues.

— Doubling the capacity of well located satellites leads to a much larger routing cost decrease than doubling the number of satellites.

— Integrating forward and backward flow can generate considerable savings (at least 40% on average for 100 customer instances and infinite satellite capacities). This saving goes decreasing with the satellites capacities and the number of satellites but always remains significant.

---

1. These instances will be made available to the scientific community for further research

Given the complexity of the 2E-MTVRP-CSRF, there are clearly many avenues to explore in order to improve its resolution, whether exact or heuristic. In particular, the feasibility problem related to the satellite capacity with forward and backward flows is a complex scheduling problem. For future studies, different backhauling policies could be compared at the second echelon and customers service at the first echelon could be integrated to model more practices evolving on the field (Nolz et al., 2020).

CONCLUSION AND PERSPECTIVES

In this thesis we proposed theoretical models for routing problems in city logistics, integrating synchronized resources to take into account the tight constraints induced by the urban environment.

We first proposed the Vehicle Routing Problem with Delivery Options (VRPDO) to integrate alternative delivery modes. Second, we proposed the 2-Echelon Multiple-Trip Vehicle Routing Problem with Capacitated Satellite and Reverse Flow (2E-MTVRP-CSRF). This problem integrates linehaul and backhauls in a 2-echelon system where satellites have limited storage capacity. To solve these problems, we developed a Large Neighborhood Search framework based on small and fast iterations: the Small and Large Neighborhood Search (SLNS). Furthermore, we hybridized this framework with MIP solved by commercial solvers and dynamic programming, thus showing that matheuristics are relevant for vehicle routing problems with synchronized resources.

4.1 Applicative contributions and perspectives

Figure 4.1 illustrates the managerial contributions and perspectives of this thesis: multiple delivery options, 2-echelon system with capacitated satellites, and multi-modality.

The utilization of multiple delivery options per customer was explored in Chapter 1. The Vehicle Routing Problem with Delivery Option (VRPDO) integrates the delivery in private and shared locations while taking into account customer's time windows and preferences, as well as the capacity of shared delivery locations. Experiments performed on randomly generated instances suggest that this can reduce the routing cost by 30% and improve the quality of service compared to the single-option per customer usual policy.

In Chapter 3, we defined the 2-Echelons Multi-Trip Vehicle Routing Problem with Capacitated Satellites and Reverse Flows (2E-MTVRP-CSRF). This problem integrates the collection and the delivery of goods in a two-echelon system with limited capacity satellites. Experiments performed on randomly generated instances with medium size satellites suggest that the integration of the linehaul and backhaul flows in this context lead to a 43% decrease of the routing costs.

The VRPDO and the 2E-MTVRP-CSRF could be firstly combined to further reduce the costs and take advantage of the many opportunities arising in cities. This integration could then be further strengthened by using multi-modality. That is to say, we could use different types of vehicles at the second echelon, from cargo-bikes to light electric vans through sidewalk drones or walking trips. It would also be possible to associate the first-echelon vehicles with scheduled lines networks accessing the city center, such as public transport or waterways. Such a complete integration of city modalities would be beneficial for both the delivery company and the residents. In addition, it would be very challenging to consider time dependency and robustness to improve the realizability of such a system.

---

[1] Designed by Quentin Le Gouvello De La Porte using assets distributed by `http://thenounproject.com` under Creative Commons license created by RROOK, mohkamil, jeff, GreenHill, Jony, indra anis, Teuku Syahrizal

Figure 4.1 – Integration and perspectives of the managerial contributions [1]

4.2   Methodological contributions and perspectives

Figure 4.2 presents the methodological contributions and perspectives of this thesis: Small and Large Neighborhood Search (SLNS), and Iterative Two-Stage Heuristic (ITSH).

We began Chapter 1 with a rigorous comparison of many operators developed over the first year of the thesis for LNS heuristics. We studied their impact on the quality of the final solution as well as the redundancy between them in order to propose an efficient set of operators. After many rounds of tests, we developed a configuration of LNS orchestrating small and large destructions: SLNS. This framework mostly performs small and local destructions allowing for fast repair. Many of these iterations can be quickly performed to successively improve parts of the solution. In addition, large destructions are performed for exploration if the solution was not improved by the preceding iterations. This new configuration of LNS was proven to be very efficient on the VRPDO compared to previously developed configurations of LNS.

In Chapter 2 we studied the hybridization of SLNS with exact methods: a Set Partitioning Problem (SPP) solved by a MILP solver, and the Balas-Simonetti neighborhood used with dynamic programming. First, we worked on the utilization of a SPP formulation of the VRPDO to recombine routes generated by SLNS. To get the best of this method, we developed an adaptive criterion to decide on when to call the solver. It is based on past performance of SLNS and those of the solver during the run. Second, we tried to improve solutions with dynamic programming by adapting the Balas-Simonetti (BS) neighborhood to the VRPDO. As for the SPP, we designed an adaptive layer to decide on when to use this technology. The strengths and weaknesses of each component of the newly developed SLNS+SPP+BS were then evaluated on multiple generalized vehicle routing problems with time windows. We concluded that SLNS was already very efficient on its own, that SPP could improve its results on hard instances, and that BS could also be useful on some specific instances.

To solve the 2E-MTVRP-CSRF, we developed the Iterative Two-Stages Heuristic (ITSH). This method iteratively improves the second-echelon routes, the first-echelon routes, and recombines them. The first two sub-problems are solved by SLNS heuristics and the recombination phase is ensured by the solving of a trip-based MIP model. Thus, ITSH relies on the tools we previously developed. In addition, to take into account the synchronization between the two echelons and the satellites capacity in SLNS, we had to integrate algorithms from the Resource Constraint Project Scheduling Problem (RCPSP) literature. Indeed, the insertion feasibility tests are based on the PERT algorithm and the double-justification principle.

Several ideas developed during this thesis could not be fully investigated. We conclude this manuscript by enumerating these research avenues:

— Exact reconstruction with Miller-Tucker-Zemlin (MTZ) MIP models: during my stay at Mainz, we explored the possibility of a MIP-based recreate heuristic for the SLNS developed for the VRPDO. This operator was intended to repair the solution by solving an MTZ-based model of the VRPDO. Unfortunately, the time required by CPLEX to find a better solution increases non-linearly with respect to the destruction size. Thus, we did not succeed in making it more profitable than list recreate operators even for promising solutions. Nonetheless, the question of exact LNS operators is still to be explored.

— Improvement of solutions via dynamic programming through giant tour representation: during our study on the Balas-Simonetti neighborhood for the VRPDO, we tried to use it on a giant tour representation of the solutions. Thus, the dynamic programming was able to choose the delivery option for all the customers. But it proved to be too computationally requiring to be efficient. Nonetheless, we still believe that it can be a promising idea by using advanced techniques from the split literature.

— Acceptance criterion limiting solution copy: the spirit of SLNS is to greatly decrease the time spent in repairing solutions in order to perform more iterations. However, at the beginning of each iteration, the time spent to copy the solution remains constant compared to an LNS with larger destructions. Thus, the proportion of the time budget spent in copies gets significant in SLNS. We tried to develop an acceptance criterion to limit the number of solution copies, but we did not come up with something effective both limiting the copy and providing enough intensification. Consequently, the question of a method to limit the time spent in solution copy remains unsolved.

— Granular insertion schemes: when we were working on improving LNS for the VRPDO in order to tackle larger instances, we tried to limit the number of insertion evaluations. The developed LNS was modified in a granular sense. That is to say, for each customer to be inserted, only the insertions next to pre-computed nearby locations were evaluated. But this component was not easy to tune for the VRPDO. Besides, this functionality was introduced in the code much too late to be efficiently implemented. Consequently, the

preliminary results were not conclusive, but we still believe that such an idea could be improving for some problems.

— Improving the main loop in ITSH: in the decomposition method developed for the 2E-MTVRP-CSRF, the method always solves the second-echelon, the first one, and possibly the MIP. We think that it can be of some use to dynamically adapt the speed of the communication between the first two subproblems to better balance intensification and diversification. We could think of a *local-search drop-like* procedure that would remove first-echelon visits or even satellites from the solution, quickly solve the second-echelon and the first-echelon, in order to generate a diversity of routes for the MIP.

— Adapting the cuts for the 2E-VRPTW of Marques et al. (2020b) to the 2E-MTVRP-CSRF: in his thesis some new cuts for the 2E-VRPTW proved to be very effective. It could significantly improve the performance of the MIP in our decomposition method to adapt them to the 2E-MTVRP-CSRF.

— Exactly build the first echelon in the MIP for the 2E-MTVRP-CSRF: in our decomposition method there are often a small number of first-echelon visits at satellites. However, when there are too many visits, the LNSs has difficulties to delete some. Thus, we thought about using an MTZ-based formulation to construct the first-echelon in the MIP, instead of using a column based-model.

— Dynamic definition of the FED in ITSH: in our 2E-MTVRP-CSRF results we observe a correlation intertwining the gap between the cost of the solution with and without split at the first echelon and the gap between the cost of the best-found solution and the average cost. An interpretation is that the former gap indicates a difficult instance. The capacity of our method to deal with this form of complexity could be improved by allowing the LNS for the first-echelon to use split delivery in a restricted way. We thought about defining FED with wisely chosen splits of some demand so that LNS could be used to improve the solutions with splits computed by the MIP.
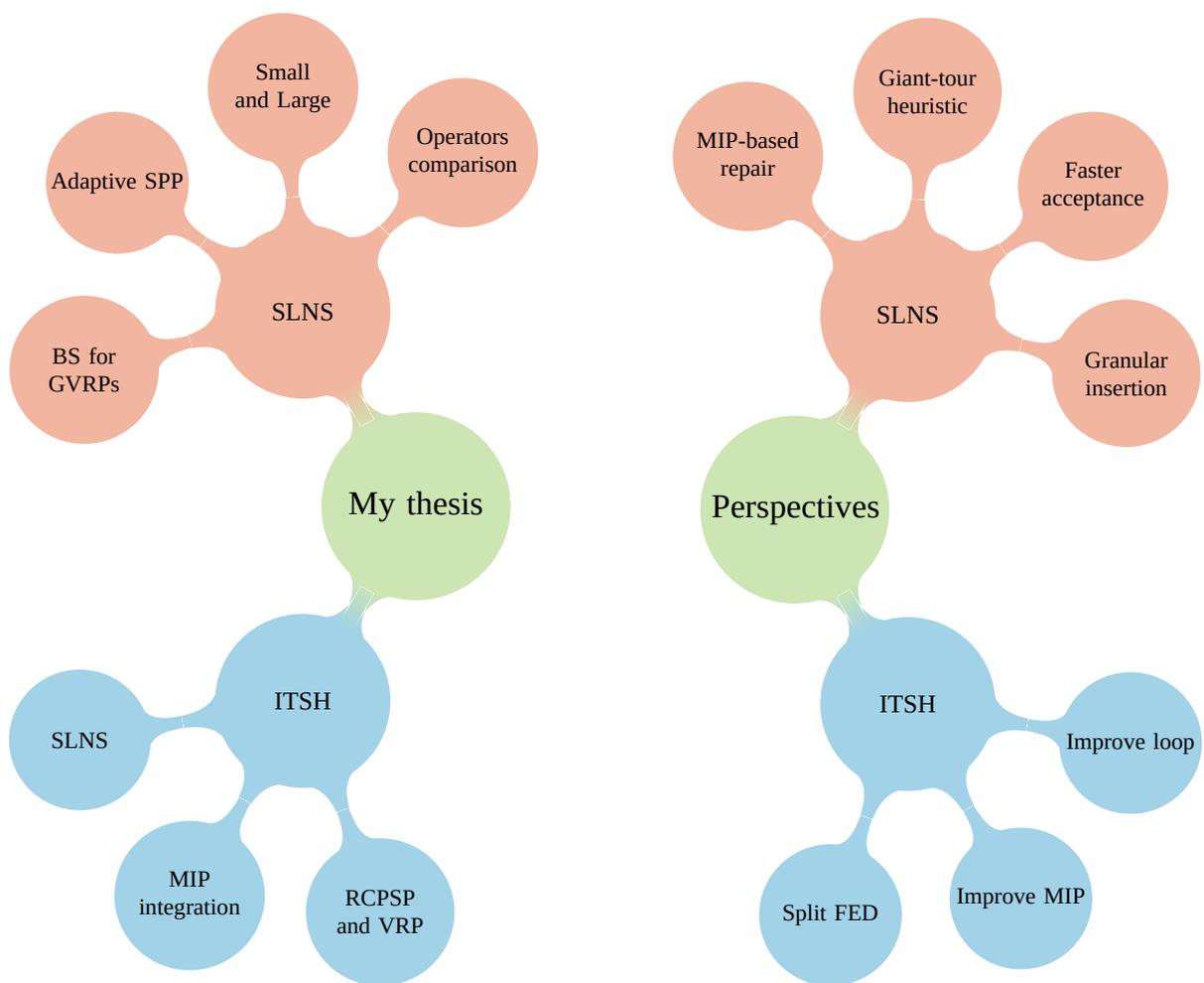
Figure 4.2 – Mind-map of the methodological contributions and perspectives

As I started this manuscript with a citation of a leading scientist of the VRP community, I would like to do the same for ending it, with this feeling expressed in Laporte (2009):

> 66 There is, however, a sense that several of the most successful metaheuristics are over-engineered and one should now attempt to produce simple and more flexible algorithms capable of handling a larger variety of constraints, even if this were to translate into a small loss in accuracy. 99

Indeed, the development of a state-of-the-art method for the generalized vehicle routing problems based on easy-to-implement list heuristics goes in this direction. Likewise, the proposed hybridizations for SLNS are based on simple adaptive layers and can be performed by existing software. But, on the contrary, facing the complexity of the 2E-MTVRP-CSRF, we developed a complex method, thus, improving it through simplifications is a major research area to interest companies in integrating our algorithms for city logistics optimization.

BIBLIOGRAPHY

H. Murat Afsar, Christian Prins, and Andréa Cynthia Santos. Exact and heuristic algorithms for solving the generalized vehicle routing problem with flexible fleet size. *International Transactions in Operational Research*, 21(1), January 2014. doi:10.1111/itor.12041.

Niels Agatz, Ann Campbell, Moritz Fleischmann, and Martin Savelsbergh. Time slot management in attended home delivery. *Erasmus Research Institute of Management*, 45, 01 2008. doi:10.2307/23018537.

Somayeh Allahyari, Majid Salari, and Daniele Vigo. A hybrid metaheuristic algorithm for the multi-depot covering tour vehicle routing problem. *European Journal of Operational Research*, 242(3), May 2015. doi:10.1016/j.ejor.2014.10.048.

Julian Allen, Maja Piecyk, and Marzena Piotrowska. Analysis of the parcels market and parcel carriers' operations in the UK. Technical report, University of Westminster, 2016. URL `http://www.ftc2050.com/reports/westminster_parcels_final_Dec_2016.pdf`.

Youcef Amarouche, Rym Nesrine Guibadj, and Aziz Moukrim. A neighborhood search and set cover hybrid heuristic for the two-echelon vehicle routing problem. In Ralf Borndörfer and Sabine Storandt, editors, *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, (ATMOS 2018)*, volume 65, Helsinki, Finland, 2018. doi:10.4230/OASIcs.ATMOS.2018.11.

Alexandra Anderluh, Vera C. Hemmelmayr, and Pamela C. Nolz. Synchronizing vans and cargo bikes in a city distribution network. *Central European Journal of Operations Research*, 25 (2), 2017. doi:10.1007/s10100-016-0441-z.

Alexandra Anderluh, Rune Larsen, Vera C. Hemmelmayr, and Pamela C. Nolz. Impact of travel time uncertainties on the solution cost of a two-echelon vehicle routing problem with synchronization. *Flexible Services and Manufacturing Journal*, 32(4), 2020. doi:10.1007/s10696-019-09351-w.

Claudia Archetti and M. Grazia Speranza. A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, 2(4), November 2014. doi:10.1007/s13675-014-0030-7.

Claudia Archetti and Maria Grazia Speranza. The Split Delivery Vehicle Routing Problem: A Survey. In Bruce Golden, S. Raghavan, and Edward Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43, pages 103–122. Springer US, Boston, MA, 2008. ISBN 978-0-387-77777-1 978-0-387-77778-8. doi:10.1007/978-0-387-77778-8_5.

Claudia Archetti, M. Grazia Speranza, and Daniele Vigo. Chapter 10: Vehicle Routing Problems with Profits. In Paolo Toth and Daniele Vigo, editors, *Vehicle Routing*. Society for Industrial and Applied Mathematics, Philadelphia, PA, November 2014. doi:10.1137/1.9781611973594.ch10.

Egon Balas. New classes of efficiently solvable generalized Traveling Salesman Problems. *Annals of Operations Research*, 86, 1999. doi:10.1023/A:1018939709890.

Egon Balas and Neil Simonetti. Linear Time Dynamic-Programming Algorithms for New Classes of Restricted TSPs: A Computational Study. *INFORMS Journal on Computing*, 13 (1), February 2001. doi:10.1287/ijoc.13.1.56.9748.

Tolga Bektaş, Teodor Gabriel Crainic, and Tom Van Woensel. From managing urban freight to smart city logistics networks. In *Network Design and Optimization for Smart Cities*, Series on Computers and Operations Research. World Scientific, 2017. doi:10.1142/9789813200012_0007.

Tolga Bektaş, Güneş Erdoğan, and Stefan Røpke. Formulations and Branch-and-Cut Algorithms for the Generalized Vehicle Routing Problem. *Transportation Science*, 45(3):299–316, August 2011. doi:10.1287/trsc.1100.0352.

Onder Belgin, Ismail Karaoglan, and Fulya Altiparmak. Two-echelon vehicle routing problem with simultaneous pickup and delivery: Mathematical model and heuristic approach. *Computers & Industrial Engineering*, 115, 2018. doi:10.1016/j.cie.2017.10.032.

Slim Belhaiza, Pierre Hansen, and Gilbert Laporte. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. *Computers & Operations Research*, 52, December 2014. doi:10.1016/j.cor.2013.08.010.

154

Slim Belhaiza, Rym M'Hallah, and Ghassen Ben Brahim. A new Hybrid Genetic Variable Neighborhood search heuristic for the Vehicle Routing Problem with Multiple Time Windows. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, Donostia, San Sebastián, Spain, June 2017. IEEE. doi:10.1109/CEC.2017.7969457.

Raquel Bernardino and Ana Paias. Metaheuristics based on decision hierarchies for the traveling purchaser problem. *International Transactions in Operational Research*, 25(4), July 2018. doi:10.1111/itor.12330.

Ulrich Breunig, Verena Schmid, Richard F. Hartl, and Thibaut Vidal. A large neighbourhood based heuristic for two-echelon routing problems. *Computers & Operations Research*, 76, 2016. doi:10.1016/j.cor.2016.06.014.

Bruno P. Bruck and Manuel Iori. Non-elementary formulations for single vehicle routing problems with pickups and deliveries. *Operations Research*, 65(6), 2017. doi:10.1287/opre.2017.1639.

Olli Bräysy and Michel Gendreau. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 39(1), February 2005a. doi:10.1287/trsc.1030.0056.

Olli Bräysy and Michel Gendreau. Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. *Transportation Science*, 39(1), February 2005b. doi:10.1287/trsc.1030.0057.

Giovanni Buzzega and Stefano Novellani. Last mile deliveries with lockers: formulations and algorithms. preprint, Universita degli Studi di Modena e Reggio Emilia, May 2021. URL https://www.researchsquare.com/article/rs-410932/v1.

J. Arturo Castillo-Salazar, Dario Landa-Silva, and Rong Qu. Workforce scheduling and routing problems: literature survey and computational study. *Annals of Operations Research*, 239(1), 2016. doi:10.1007/s10479-014-1687-2.

Jan Christiaens and Greet Vanden Berghe. Slack Induction by String Removals for Vehicle Routing Problems. *Transportation Science*, January 2020. doi:10.1287/trsc.2019.0914.

Teodor Gabriel Crainic, Nicoletta Ricciardi, and Giovanni Storchi. Models for Evaluating and Planning City Logistics Systems. *Transportation Science*, 43(4), November 2009. doi:10.1287/trsc.1090.0279.

Teodor Gabriel Crainic, Fausto Errico, Walter Rei, and Nicoletta Ricciardi. Integrating c2e and c2c Traffic into City Logistics Planning. *Procedia - Social and Behavioral Sciences*, 39, January 2012. doi:10.1016/j.sbspro.2012.03.090.

Rosario Cuda, Gianfranco Guastaroba, and Maria Grazia Speranza. A survey on two-echelon routing problems. *Computers & Operations Research*, 55, 2015. doi:10.1016/j.cor.2014.06.008.

Timothy Curtois, Dario Landa-Silva, Yi Qu, and Wasakorn Laesanklang. Large neighbourhood search with adaptive guided ejection search for the pickup and delivery problem with time windows. *EURO Journal on Transportation and Logistics*, 7(2), June 2018. doi:10.1007/s13676-017-0115-6.

G. B. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6(1), October 1959. doi:10.1287/mnsc.6.1.80.

Nico Dellaert, Fardin Dashty Saridarq, Tom van Woensel, and Teodor Gabriel Crainic. Branch-and-price–based algorithms for the two-echelon vehicle routing problem with time windows. *Transportation Science*, 53(2), 2019. doi:10.1287/trsc.2018.0844.

Nico Dellaert, Tom van Woensel, Teodor Gabriel Crainic, and Fardin Dashty Saridarq. A multi-commodity two-echelon capacitated vehicle routing problem with time windows: Model formulations and solution approach. *Computers & Operations Research*, 127, 2021. doi:10.1016/j.cor.2020.105154.

Emrah Demir, Tolga Bektaş, and Gilbert Laporte. An adaptive large neighborhood search heuristic for the Pollution-Routing Problem. *European Journal of Operational Research*, 223(2), December 2012. doi:10.1016/j.ejor.2012.06.044.

Guy Desaulniers, Oli B.G. Madsen, and Stefan Ropke. Chapter 5: The Vehicle Routing Problem with Time Windows. In Paolo Toth and Daniele Vigo, editors, *Vehicle Routing*. Society for Industrial and Applied Mathematics, Philadelphia, PA, November 2014. doi:10.1137/1.9781611973594.ch5.

Guy Desaulniers, Fausto Errico, Stefan Irnich, and Michael Schneider. Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6), 2016. doi:10.1287/opre.2016.1535.

Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2), 2002. doi:10.1007/s101070100263.

Michael Drexl. Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints. *Transportation Science*, 46(3), August 2012. doi:10.1287/trsc.1110.0400.

Gunter Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational physics*, 104(1), 1993. doi:10.1006/jcph.1993.1010.

Gunter Dueck and Tobias Scheuer. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1), 1990. doi:10.1016/0021-9991(90)90201-B.

Dorian Dumez, Fabien Lehuédé, and Olivier Péton. Integrating lockers and multiple delivery options in a city logistics distribution system. In *EURO 2019*, Dublin, Ireland, June 2019a. URL `https://hal.archives-ouvertes.fr/hal-02367987`.

Dorian Dumez, Fabien Lehuédé, and Olivier Péton. A Large Neighborhood Search approach to integrate delivery options in last mile delivery. In *VeRoLog 2019 : seventh annual workshop of the EURO Working Group on Vehicle Routing and Logistics Optimization*, Sevilla, Spain, June 2019b. URL `https://hal.archives-ouvertes.fr/hal-02285392`.

Dorian Dumez, Irnich Stefan, Fabien Lehuédé, Katharina Olkis, Olivier Péton, and Christian Tilk. Large Neighborhood Search with set partitioning and dynamic programming for the VRP with Delivery Options. In *SynchroTrans 2019 : Second International Workshop on Synchronization in Transport*, Nantes, France, September 2019c. URL `https://hal.archives-ouvertes.fr/hal-02285324`.

Dorian Dumez, Katharina Olkis, Stefan Irnich, Fabien Lehuédé, Olivier Péton, and Christian Tilk. Renforcements de la recherche à voisinage large pour les problèmes de tournées de véhicules généralisés. In *ROADEF 2020*, February 2020. URL `https://hal.archives-ouvertes.fr/hal-02491381`.

Dorian Dumez, Fabien Lehuédé, and Olivier Péton. A large neighborhood search approach to the vehicle routing problem with delivery options. *Transportation Research Part B: Methodological*, 144, February 2021a. doi:10.1016/j.trb.2020.11.012.

Dorian Dumez, Christian Tilk, Stefan Irnich, Fabien Lehuédé, Olkis Katharina, and Olivier Péton. A matheuristic for a 2-echelon-vehicle routing problem with satellite capacities and reverse flows. working paper or preprint, June 2021b.

Dorian Dumez, Christian Tilk, Stefan Irnich, Fabien Lehuédé, and Olivier Péton. Hybridizing Large Neighborhood Search and Exact Methods for Generalized Vehicle Routing Problems with Time Windows. *EURO Journal on Transportation and Logistics*, May 2021c. doi:10.1016/j.ejtl.2021.100040.

Nizar El Hachemi, Michel Gendreau, and Louis-Martin Rousseau. A hybrid constraint programming approach to the log-truck scheduling problem. *Annals of Operations Research*, 184(1), 2011. doi:10.1007/s10479-010-0698-x.

Nizar El Hachemi, Michel Gendreau, and Louis-Martin Rousseau. A heuristic to solve the synchronized log-truck scheduling problem. *Computers & Operations Research*, 40(3), 2013. doi:10.1016/j.cor.2011.02.002.

Ralf Elbert and Christian Friedrich. Urban consolidation and cargo bikes: a simulation study. *Transportation Research Procedia*, 48, 2020. doi:10.1016/j.trpro.2020.08.051.

Daniela Favaretto, Elena Moretti, and Paola Pellegrini. Ant colony system for a VRP with multiple time windows and multiple visits. *Journal of Interdisciplinary Mathematics*, 10(2), April 2007. doi:10.1080/09720502.2007.10700491.

Dominique Feillet, Michel Gendreau, and Louis-Martin Rousseau. New Refinements for the Solution of Vehicle Routing Problems with Branch and Price. *INFOR*, 45(4), November 2007. doi:10.3138/infor.45.4.239.

Christian Fikar and Patrick Hirsch. Home health care routing and scheduling: A review. *Computers & Operations Research*, 77, 2017. doi:10.1016/j.cor.2016.07.019.

Martina Fischetti and Matteo Fischetti. Matheuristics. In Rafael Martí, Panos M. Pardalos, and Mauricio G. C. Resende, editors, *Handbook of Heuristics*. Springer International Publishing, 2018. doi:10.1007/978-3-319-07124-4_14.

Bernhard Fleischmann. The vehicle routing problem with multiple use of the vehicles. *Forschungsbericht Fachbereich Wirtschaftswissenschaften, Universität Hamburg*, January 1990. URL `https://www.researchgate.net/publication/221704650_The_vehicle_routing_problem_with_multiple_use_of_vehicles`.

Alexandre M. Florio, Dominique Feillet, and Richard F. Hartl. The delivery problem: Optimizing hit rates in e-commerce deliveries. *Transportation Research Part B: Methodological*, 117, 2018. doi:10.1016/j.trb.2018.09.011.

B A Foster. An Integer Programming Approach to the Vehicle Scheduling Problem. *Taylor & Francis*, 27(2), 1976. doi:10.1057/jors.1976.63.

Aurélien Froger, Jorge E Mendoza, Ola Jabali, and Gilbert Laporte. A Matheuristic for the Electric Vehicle Routing Problem with Capacitated Charging Stations. Technical Report CIRRELT-2017-31, CIRRELT, 2017. URL `https://hal.archives-ouvertes.fr/hal-01559524`.

Gosia Furmanik. How much does retail space cost? *realla*, 2019. URL `https://blog.realla.co.uk/how-much-does-retail-space-cost`.

Hermann Gehring and Jörg Homberger. A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows. In *EUROGEN99*, 1999. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.617.1364&rep=rep1&type=pdf`.

Michel Gendreau, Daniele Manerba, and Renata Mansini. The multi-vehicle traveling purchaser problem with pairwise incompatibility constraints and unitary demands: A branch-and-price approach. *European Journal of Operational Research*, 248(1), January 2016. doi:10.1016/j.ejor.2015.06.073.

Gianpaolo Ghiani and Gennaro Improta. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research*, 122(1), 2000. doi:10.1016/S0377-2217(99)00073-9.

Fred W. Glover and Gary A. Kochenberger, editors. *Handbook of Metaheuristics*. International Series in Operations Research & Management Science. Springer US, 2003. doi:10.1007/b101874.

Jesus Gonzalez-Feliu, Guido Perboli, Roberto Tadei, and Daniele Vigo. The two-echelon capacitated vehicle routing problem. working paper, halshs-00879447, 2008. URL `https://halshs.archives-ouvertes.fr/halshs-00879447`.

159

Jasmin Grabenschweiger, Karl F. Doerner, Richard F. Hartl, and Martin W. P. Savelsbergh. The vehicle routing problem with heterogeneous locker boxes. *Central European Journal of Operations Research*, January 2021. doi:10.1007/s10100-020-00725-2.

Philippe Grangier, Michel Gendreau, Fabien Lehuédé, and Louis-Martin Rousseau. An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *European Journal of Operational Research*, 254(1), October 2016. doi:10.1016/j.ejor.2016.03.040.

Philippe Grangier, Michel Gendreau, Fabien Lehuédé, and Louis-Martin Rousseau. A matheuristic based on large neighborhood search for the vehicle routing problem with cross-docking. *Computers & Operations Research*, 84, August 2017. doi:10.1016/j.cor.2017.03.004.

Philippe Grangier, Michel Gendreau, Fabien Lehuédé, and Louis-Martin Rousseau. A lns and branch-and-check approach for a vrp with cross-docking and ressource synchronization. In *INFORMS TSL Workshop on "E-Commerce and Urban Logistics"*, 2018. URL `https://hal.archives-ouvertes.fr/hal-01925325/document`.

Philippe Grangier, Michel Gendreau, Fabien Lehuédé, and Louis-Martin Rousseau. The vehicle routing problem with cross-docking and resource constraints. *Journal of Heuristics*, 2019. doi:10.1007/s10732-019-09423-y.

Axel Grimault, Nathalie Bostel, and Fabien Lehuédé. An adaptive large neighborhood search for the full truckload pickup and delivery problem with resource synchronization. *Computers & Operations Research*, 88, December 2017. doi:10.1016/j.cor.2017.06.012.

Timo Gschwind and Michael Drexl. Adaptive Large Neighborhood Search with a Constant-Time Feasibility Test for the Dial-a-Ride Problem. *Transportation Science*, 53(2), March 2019. doi:10.1287/trsc.2018.0837.

Sylvie Gélinas, Martin Desrochers, Jacques Desrosiers, and Marius M. Solomon. A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61(1), December 1995. doi:10.1007/BF02098283.

Pierre Hansen, Nenad Mladenović, and José A. Moreno Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1), 2009. doi:10.1007/s10479-009-0657-6.

Stefan Helber and Florian Sahling. A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics*, 123(2), 2010. doi:10.1016/j.ijpe.2009.08.022.

Vera C. Hemmelmayr, Jean-François Cordeau, and Teodor Gabriel Crainic. An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics. *Computers & Operations Research*, 39(12), December 2012. doi:10.1016/j.cor.2012.04.007.

Christoph Hempsch and Stefan Irnich. Vehicle Routing Problems with Inter-Tour Resource Constraints. In Bruce Golden, S. Raghavan, and Edward Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges,* volume 43. Springer US, Boston, MA, 2008. doi:10.1007/978-0-387-77778-8_19.

Timo Hintsch and Stefan Irnich. Large multiple neighborhood search for the clustered vehicle-routing problem. *European Journal of Operational Research*, 270(1), October 2018. doi:10.1016/j.ejor.2018.02.056.

Maaike Hoogeboom, Wout Dullaert, David Lai, and Daniele Vigo. Efficient Neighborhood Evaluations for the Vehicle Routing Problem with Multiple Time Windows. *Transportation Science,* January 2020. doi:10.1287/trsc.2019.0912.

Qian Hu and Andrew Lim. An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 232(2), January 2014. doi:10.1016/j.ejor.2013.06.011.

IBM. *CPLEX*, 2018. https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer.

Stefan Irnich. Resource extension functions: properties, inversion, and generalization to segments. *OR Spectrum*, 30(1), 2008a. doi:10.1007/s00291-007-0083-6.

Stefan Irnich. A Unified Modeling and Solution Framework for Vehicle Routing and Local Search-Based Metaheuristics. *INFORMS Journal on Computing*, 20(2), May 2008b. doi:10.1287/ijoc.1070.0239.

Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors, *Column Generation.* Springer-Verlag, 2005. doi:10.1007/0-387-25486-2_2.

Stefan Irnich, Paolo Toth, and Daniele Vigo. The family of vehicle routing problems. In *Vehicle Routing*, chapter 1. Society for Industrial & Applied Mathematics (SIAM), 2014. doi:10.1137/1.9781611973594.ch1.

Peter Jackson, Fran Walsh, and Philippa Boyens. The lord of the rings: The return of the king, 2003.

Wanchen Jie, Jun Yang, Min Zhang, and Yongxi Huang. The two-echelon capacitated electric vehicle routing problem with battery swapping stations: Formulation and efficient methodology. *European Journal of Operational Research*, 272(3), 2019. doi:10.1016/j.ejor.2018.07.002.

Manel Kammoun, Houda Derbel, Mostapha Ratli, and Bassem Jarboui. An integration of mixed VND and VNS: the case of the multivehicle covering tour problem: An integration of mixed VND and VNS. *International Transactions in Operational Research*, 24(3), May 2017. doi:10.1111/itor.12355.

Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598), May 1983. doi:10.1126/science.220.4598.671.

Gilbert Laporte. Fifty Years of Vehicle Routing. *Transportation Science*, 43(4), November 2009. doi:10.1287/trsc.1090.0301.

Rune Larsen and Dario Pacino. Fast delta evaluation for the Vehicle Routing Problem with Multiple Time Windows. Technical report, Technical University of Denmark, May 2019. URL http://arxiv.org/abs/1905.04114. arXiv: 1905.04114.

Hongqi Li, Yinying Liu, Xiaorong Jian, and Yingrong Lu. The two-echelon distribution system considering the real-time transshipment capacity varying. *Transportation Research Part B: Methodological*, 110(2), 2018. doi:10.1016/j.trb.2018.02.015.

Hongqi Li, Yinying Liu, Kaihang Chen, and Qingfeng Lin. The two-echelon city logistics system with on-street satellites. *Computers & Industrial Engineering*, 139, 2020. doi:10.1016/j.cie.2018.12.024.

Augustin Lombard, Simon Tamayo-Giraldo, and Frédéric Fontane. Vehicle Routing Problem with Roaming Delivery Locations and Stochastic Travel Times (VRPRDL-S). *Transportation Research Procedia*, 30, 2018. doi:10.1016/j.trpro.2018.09.019.

Leonardo Lozano and Andrés L. Medaglia. On an exact method for the constrained shortest path problem. *Computers & Operations Research*, 40(1), 2013. doi:10.1016/j.cor.2012.07.008.

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 2016. doi:10.1016/j.orp.2016.09.002.

Simona Mancini. A combined multistart random constructive heuristic and set partitioning based formulation for the vehicle routing problem with time dependent travel times. *Computers & Operations Research*, 88, December 2017. doi:10.1016/j.cor.2017.06.021.

Simona Mancini and Margaretha Gansterer. Vehicle routing with private and shared delivery locations. *Computers & Operations Research*, 133, September 2021. doi:10.1016/j.cor.2021.105361.

Guillaume Marques. *Two-echelon vehicle routing problems in city logistics: Approaches based on exact methods of mathematical optimization.* PhD thesis, 2020.

Guillaume Marques, Ruslan Sadykov, Jean-Christophe Deschamps, and Rémy Dupas. A branch-cut-and-price approach for the single-trip and multi-trip two-echelon vehicle routing problem with time windows. Technical Report 03139799, Inria, 2020a. URL `https://hal.inria.fr/hal-03139799`.

Guillaume Marques, Ruslan Sadykov, Jean-Christophe Deschamps, and Rémy Dupas. An improved branch-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. *Computers & Operations Research*, 114, February 2020b. doi:10.1016/j.cor.2019.104833.

Renaud Masson, Fabien Lehuédé, and Olivier Péton. Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. *Operations Research Letters*, 41 (3), May 2013. doi:10.1016/j.orl.2013.01.007.

Renaud Masson, Fabien Lehuédé, and Olivier Péton. The dial-a-ride problem with transfers. *Computers & Operations Research*, 41, 2014. doi:10.1016/j.cor.2013.07.020.

Matt McFarland. Amazon now delivers to the trunk of your car. *CNN business*, 2018. URL `https://money.cnn.com/2018/04/24/technology/amazon-key-in-car-delivery-review/index.html`.

Jorge E. Mendoza and Juan G. Villegas. A multi-space sampling heuristic for the vehicle routing problem with stochastic demands. *Optimization Letters*, 7(7), October 2013. doi:10.1007/s11590-012-0555-8.

Tayeb Mhamedi, Henrik Andersson, Marilène Cherkesly, and Guy Desaulniers. A branch-price-and-cut algorithm for the two-echelon vehicle routing problem with time windows. Cahiers du GERAD G-2020-63, GERAD, HEC Montréal, Canada, 2020. URL `https://www.gerad.ca/fr/papers/G-2020-63`.

Robert Wallace Miller. *Schedule, cost, and profit control with PERT: A comprehensive guide for program management*. McGraw-Hill, 1963.

L Moccia, J-F Cordeau, and G Laporte. An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. *The Journal of the Operational Research Society*, 63(2), 2012. doi:10.1057/jors.2011.25.

Eleonora Morganti, Saskia Seidel, Corinne Blanquart, Laetitia Dablanc, and Barbara Lenz. The impact of e-commerce on final deliveries: alternative parcel delivery services in France and Germany. *Transportation Research Procedia*, 4, 2014. doi:10.1016/j.trpro.2014.11.014.

E. Moyou. Chiffre d'affaires annuel du e-commerce en france de 2005 à 2018, 2019. URL `https://fr.statista.com/statistiques/474685/chiffre-d-affaires-e-commerce-france/`.

Ferdinand Mühlbauer and Pirmin Fontaine. A parallelised large neighbourhood search heuristic for the asymmetric two-echelon vehicle routing problem with swap containers for cargo-bicycles. *European Journal of Operational Research*, 289(2), 2021. doi:10.1016/j.ejor.2020.07.034.

Jesús Muñuzuri, Rafael Grosso, Pablo Cortés, and José Guadix. Estimating the extra costs imposed on delivery vehicles using access time windows in a city. *Computers, Environment and Urban Systems*, 41, 2013. doi:10.1016/j.compenvurbsys.2012.05.005.

Yuichi Nagata and Olli Bräysy. A powerful route minimization heuristic for the vehicle routing problem with time windows. *Operations Research Letters*, 37(5), September 2009. doi:10.1016/j.orl.2009.04.006.

Pamela C. Nolz, Nabil Absi, Diego Cattaruzza, and Dominique Feillet. Two-echelon distribution with a single capacitated city hub. *EURO Journal on Transportation and Logistics*, 9(3), 2020. doi:10.1016/j.ejtl.2020.100015.

Gizem Ozbaygin, Oya Ekin Karasan, Martin Savelsbergh, and Hande Yaman. A branch-and-price algorithm for the vehicle routing problem with roaming delivery locations. *Transportation Research Part B: Methodological*, 100, June 2017. doi:10.1016/j.trb.2017.02.003.

Rosario Paradiso, Roberto Roberti, Demetrio Laganá, and Wout Dullaert. An exact solution framework for multitrip vehicle-routing problems with time windows. *Operations Research*, 68(1), 2020. doi:10.1287/opre.2019.1874.

Dimitris C. Paraskevopoulos, Gilbert Laporte, Panagiotis P. Repoussis, and Christos D. Tarantilis. Resource constrained routing and scheduling: Review and research prospects. *European Journal of Operational Research*, 263(3), December 2017. doi:10.1016/j.ejor.2017.05.035.

Sophie N. Parragh and Verena Schmid. Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, 40(1), January 2013. doi:10.1016/j.cor.2012.08.004.

PassMark-Software. CPU benchmarks, 2020. `https://www.cpubenchmark.net`.

Guido Perboli, Roberto Tadei, and Edoardo Fadda. New valid inequalities for the two-echelon capacitated vehicle routing problem. *Electronic Notes in Discrete Mathematics*, 64, 2018. doi:10.1016/j.endm.2018.01.009.

David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8), August 2007. doi:10.1016/j.cor.2005.09.012.

David Pisinger and Stefan Ropke. Large Neighborhood Search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 272. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-319-91086-4_4.

Eric Prescott-Gagnon, Guy Desaulniers, and Louis-Martin Rousseau. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, 54(4), December 2009. doi:10.1002/net.20332.

Olivier Péton, Dorian Dumez, and Fabien Lehuédé. Vehicle Routing Problem with Alternative Delivery Options and Customer Preferences. In *IWUOR 2019 : International Workshop on Urban Operations Research*, Nagoya, Japan, July 2019. URL `https://hal.archives-ouvertes.fr/hal-02285430`.

Damián Reyes, Martin Savelsbergh, and Alejandro Toriello. Vehicle routing with roaming delivery locations. *Transportation Research Part C: Emerging Technologies*, 80, July 2017. doi:10.1016/j.trc.2017.04.003.

Giovanni Righini and Matteo Salani. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3), September 2006. doi:10.1016/j.disopt.2006.05.007.

Yves Rochat and Éric D. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1), September 1995. doi:10.1007/BF02430370.

Stefan Ropke and David Pisinger. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 40(4), November 2006a. doi:10.1287/trsc.1050.0135.

Stefan Ropke and David Pisinger. A unified heuristic for a large class of Vehicle Routing Problems with Backhauls. *European Journal of Operational Research*, 171(3), June 2006b. doi:10.1016/j.ejor.2004.09.004.

Alberto Santini, Stefan Ropke, and Lars Magnus Hvattum. A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, 24(5), October 2018. doi:10.1007/s10732-018-9377-x.

Martin Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4(1), December 1985. doi:10.1007/BF02022044.

Martin Savelsbergh and Tom Van Woensel. 50th Anniversary Invited Article—City Logistics: Challenges and Opportunities. *Transportation Science*, 50(2), March 2016. doi:10.1287/trsc.2016.0675. Publisher: INFORMS.

Martin W. P. Savelsbergh. The Vehicle Routing Problem with Time Windows: Minimizing Route Duration. *ORSA Journal on Computing*, 4(2), May 1992. doi:10.1287/ijoc.4.2.146.

Verena Schmid, Karl F Doerner, Richard F Hartl, Martin WP Savelsbergh, and Wolfgang Stoecher. A hybrid solution approach for ready-mixed concrete delivery. *Transportation Science*, 43(1), 2009. doi:10.1287/trsc.1080.0249.

Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4), 2014. doi:10.1287/trsc.2013.0490.

Gerhard Schrimpf, Johannes Schneider, Hermann Stamm-Wilbrandt, and Gunter Dueck. Record Breaking Optimization Results Using the Ruin and Recreate Principle. *Journal of Computational Physics*, 159(2), April 2000. doi:10.1006/jcph.1999.6413.

Paul Shaw. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Michael Maher, and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming — CP98*, volume 1520. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. doi:10.1007/3-540-49481-2_30.

Pawel Sitek and Jarosław Wikarek. Capacitated vehicle routing problem with pick-up and alternative delivery (CVRPPAD): model and implementation using hybrid approach. *Annals of Operations Research*, 273(1-2), February 2019. doi:10.1007/s10479-017-2722-x.

Marius M. Solomon and Jacques Desrosiers. Time Window Constrained Routing and Scheduling Problems. *Transportation Science*, 22(1), February 1988. doi:10.1287/trsc.22.1.1.

Wouter Souffriau, Pieter Vansteenwegen, Greet Vanden Berghe, and Dirk Van Oudheusden. The Multiconstraint Team Orienteering Problem with Multiple Time Windows. *Transportation Science*, 47(1), February 2013. doi:10.1287/trsc.1110.0377.

Thomas Stützle. Heuristic optimization, 2018. URL `http://iridia.ulb.ac.be/~stuetzle/Teaching/HO/`.

Anand Subramanian, Eduardo Uchoa, and Luiz Satoru Ochi. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10), October 2013a. doi:10.1016/j.cor.2013.01.013.

Anand Subramanian, Eduardo Uchoa, Artur Alves Pessoa, and Luiz Satoru Ochi. Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. *Optimization Letters*, 7(7), 2013b. doi:10.1007/s11590-012-0570-9.

Kenneth Sörensen, Marc Sevaux, and Fred Glover. A History of Metaheuristics. In Rafael Martí, Panos M. Pardalos, and Mauricio G. C. Resende, editors, *Handbook of Heuristics*. Springer International Publishing, Cham, 2018. doi:10.1007/978-3-319-07124-4_4.

Yosuke Takada, Yannan Hu, Hideki Hashimoto, and Mutsunori Yagiura. An iterated local search algorithm for the multi-vehicle covering tour problem. In *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Singapore, Singapore, December 2015. IEEE. doi:10.1109/IEEM.2015.7385846.

Oscar Tellez, Samuel Vercraene, Fabien Lehuédé, Olivier Péton, and Thibaud Monteiro. The fleet size and mix dial-a-ride problem with reconfigurable vehicle capacity. *Transportation Research Part C: Emerging Technologies*, 91, June 2018. doi:10.1016/j.trc.2018.03.020.

Christian Tilk and Stefan Irnich. Dynamic Programming for the Minimum Tour Duration Problem. *Transportation Science*, 51(2), May 2017. doi:10.1287/trsc.2015.0626.

Christian Tilk, Katharina Olkis, and Stefan Irnich. The Last-mile Vehicle Routing Problem with Delivery Options. Technical Report LM-2020-06, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany, May 2020. URL `https://download.uni-mainz.de/RePEc/pdf/Discussion_Paper_2017.pdf`.

Túlio A.M. Toffolo, Thibaut Vidal, and Tony Wauters. Heuristics for vehicle routing problems: Sequence or set optimization? *Computers & Operations Research*, 105, May 2019. doi:10.1016/j.cor.2018.12.023.

Paolo Toth and Daniele Vigo, editors. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, January 2002. doi:10.1137/1.9780898718515.

Paolo Toth and Daniele Vigo. The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *INFORMS Journal on Computing*, 15(4), November 2003. doi:10.1287/ijoc.15.4.333.24890.

Fabien Tricoire, Martin Romauch, Karl F. Doerner, and Richard F. Hartl. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, 37(2), February 2010. doi:10.1016/j.cor.2009.05.012.

Renata Turkeš, Kenneth Sörensen, and Lars Magnus Hvattum. Meta-analysis of metaheuristics: Quantifying the effect of adaptiveness in adaptive large neighborhood search. *European Journal of Operational Research*, November 2020. doi:10.1016/j.ejor.2020.10.045.

Vicente Valls, Francisco Ballestín, and Sacramento Quintanilla. Justification and RCPSP: A technique that pays. *European Journal of Operational Research*, 165(2), 2005. doi:10.1016/j.ejor.2004.04.008.

Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1), February 2011. doi:10.1016/j.ejor.2010.03.045.

Leticia Vargas, Nicolas Jozefowiez, and Sandra Ulrich Ngueveu. A Selector Operator-Based Adaptive Large Neighborhood Search for the Covering Tour Problem. In Clarisse Dhaenens, Laetitia Jourdan, and Marie-Eléonore Marmion, editors, *Learning and Intelligent Optimization*, volume 8994. Springer International Publishing, Cham, 2015. doi:10.1007/978-3-319-19084-6_16.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1), January 2013. doi:10.1016/j.cor.2012.07.018.

Kangzhou Wang, Yeming Shao, and Weihua Zhou. Matheuristic for a two-echelon capacitated vehicle routing problem with environmental considerations in city logistics service. *Transportation Research Part D: Transport and Environment*, 57, 2017. doi:10.1016/j.trd.2017.09.018.

Jerome D. Wiest. Some properties of schedules for large projects with limited resources. *Operations Research*, 12(3), 1964. doi:10.1287/opre.12.3.395.

Frank Wilcoxon. Individual Comparisons by Ranking Methods. In Samuel Kotz and Norman L. Johnson, editors, *Breakthroughs in Statistics: Methodology and Distribution*, Springer Series in Statistics. Springer, New York, NY, 1992. doi:10.1007/978-1-4612-4380-9_16.

Richard T. Wong. Vehicle routing for small package delivery and pickup services. In *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer US, 2008. doi:10.1007/978-0-387-77778-8_21.

Yuan Yuan. *Modèles et Algorithmes pour les Problèmes de Livraison du Dernier Kilomètre avec Plusieurs Options d'Expédition*. These de doctorat, Ecole centrale de Lille, October 2019. URL `http://www.theses.fr/2019ECLI0011`.

Yuan Yuan, Diego Cattaruzza, Maxime Ogier, Cyriaque Rousselot, and Frédéric Semet. Mixed integer programming formulations for the generalized traveling salesman problem with time windows. *4OR*, October 2020a. ISSN 1619-4500, 1614-2411. doi:10.1007/s10288-020-00461-y.

Yuan Yuan, Diego Cattaruzza, Maxime Ogier, and Frédéric Semet. A branch-and-cut algorithm for the generalized traveling salesman problem with time windows. *European Journal of Operational Research*, 286(3), November 2020b. doi:10.1016/j.ejor.2020.04.024.

Yuan Yuan, Diego Cattaruzza, Maxime Ogier, Frédéric Semet, and Daniele Vigo. A column generation based heuristic for the generalized vehicle routing problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, 152:102391, August 2021. ISSN 13665545. doi:10.1016/j.tre.2021.102391.

Umman Mahir Yıldırım and Bülent Çatay. An Ant Colony-Based Matheuristic Approach for Solving a Class of Vehicle Routing Problems. In Francesco Corman, Stefan Voß, and Rudy R. Negenborn, editors, *Computational Logistics*, volume 9335. Springer International Publishing, Cham, 2015. doi:10.1007/978-3-319-24264-4_8.

Lin Zhou, Roberto Baldacci, Daniele Vigo, and Xu Wang. A Multi-Depot Two-Echelon Vehicle Routing Problem with Delivery Options Arising in the Last Mile Distribution. *European Journal of Operational Research*, 265(2), March 2018. doi:10.1016/j.ejor.2017.08.011.

Appendices

DETAILED RESULTS ON VEHICLE ROUTING PROBLEMS WITH TIME WINDOWS

A.1 Detailed results on the VRPDO instances

Instance	LNS		SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS	
	Route	Cost	Route	Cost	Route	Cost	Route	Cost	Route	Cost
U_100_1	11	482.07	11	481.82	11	482.77	11	481.82	11	479.97
U_100_2	10	682.81	10	690.49	10	703.56	10	672.93	10	691.13
U_100_3	11	629.93	11	638.34	11	637.30	11	631.51	11	630.02
U_100_4	10	594.53	10	612.78	10	614.97	10	607.13	10	609.72
U_100_5	10	673.31	10	680.54	10	701.20	10	673.31	10	673.31
U_100_6	10	598.61	10	592.42	10	601.70	10	598.61	10	598.61
U_100_7	11	747.26	11	747.05	11	749.35	11	763.74	11	763.64
U_100_8	11	844.42	11	856.52	11	852.73	11	843.29	11	845.89
U_100_9	10	677.69	10	683.95	10	693.10	10	680.25	10	682.76
U_100_10	11	572.75	11	574.63	11	575.64	11	575.62	11	573.20
V_100_1	11	647.57	11	654.42	11	647.57	11	647.57	11	647.57
V_100_2	10	863.15	10	857.68	10	855.53	10	851.12	10	848.57
V_100_3	10	696.71	10	707.80	10	700.69	10	686.65	10	706.33
V_100_4	10	593.82	10	597.02	10	596.84	10	594.61	10	594.61
V_100_5	10	768.99	10	778.42	10	790.63	10	771.77	10	802.72
V_100_6	11	597.29	11	599.46	11	599.28	11	597.29	11	597.29
V_100_7	11	702.18	11	705.06	11	703.77	11	707.93	11	703.34
V_100_8	11	744.85	11	749.42	11	747.44	11	750.10	11	745.35
V_100_9	10	786.05	10	811.91	10	805.64	10	804.28	10	800.96
V_100_10	10	606.26	10	616.88	10	623.24	10	615.31	10	605.62
UBC_100_1	4	368.07	4	365.80	4	368.24	4	368.24	4	368.24
UBC_100_2	4	353.70	4	344.62	4	346.32	4	353.10	4	350.97
UBC_100_3	4	335.92	4	323.46	4	323.46	4	323.46	4	323.46
UBC_100_4	4	334.61	4	334.61	4	334.61	4	334.61	4	334.61
UBC_100_5	4	372.29	4	371.04	4	391.88	4	371.55	4	371.55
UBC_100_6	4	356.68	4	349.58	4	379.38	4	346.91	4	363.77
UBC_100_7	4	337.90	4	337.90	4	337.90	4	337.90	4	337.90
UBC_100_8	4	417.08	4	422.48	4	423.28	4	422.48	4	415.07
UBC_100_9	4	388.49	4	388.04	4	396.03	4	384.46	4	384.46
UBC_100_10	4	350.03	4	356.33	4	356.33	4	356.33	4	362.56

Table A.1 – Detailed results on the VRPDO instances with 100 customers in 300 seconds

Instance	LNS		SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS	
	Route	Cost	Route	Cost	Route	Cost	Route	Cost	Route	Cost
U_200_1	21	1 503.79	21	1 542.48	21	1 536.11	21	1 508.56	21	1 491.37
U_200_2	21	1 194.72	21	1 208.02	21	1 214.84	21	1 195.00	21	1 189.36
U_200_3	20	1 287.07	20	1 310.72	20	1 324.30	20	1 289.18	20	1 289.47
U_200_4	21	914.78	21	932.00	21	939.91	21	924.91	21	920.70
U_200_5	21	1 051.40	21	1 072.84	21	1 056.72	21	1 052.94	21	1 050.60
U_200_6	20	1 085.91	20	1 106.35	20	1 106.79	20	1 081.08	20	1 087.79
U_200_7	21	988.37	21	1 007.64	21	1 035.93	21	1 001.94	21	1 007.83
U_200_8	20	1 077.32	20	1 095.71	20	1 105.12	20	1 093.05	20	1 079.74
U_200_9	20	1 165.87	20	1 197.17	20	1 183.06	20	1 177.17	20	1 191.32
U_200_10	20	1 462.33	20	1 486.39	20	1 480.56	20	1 445.02	20	1 478.39
V_200_1	21	1 284.06	21	1 293.37	21	1 307.26	21	1 292.04	21	1 293.09
V_200_2	20	1 229.74	20	1 241.74	20	1 246.82	20	1 212.85	20	1 203.66
V_200_3	21	1 240.69	21	1 255.86	21	1 250.38	20	1 397.31	21	1 243.62
V_200_4	21	1 336.35	21	1 368.87	21	1 378.37	21	1 349.48	21	1 329.73
V_200_5	20	1 360.36	20	1 409.91	20	1 398.44	20	1 364.60	20	1 369.24
V_200_6	21	1 416.77	21	1 446.09	21	1 431.78	21	1 453.41	21	1 445.76
V_200_7	20	1 151.02	20	1 166.02	20	1 158.90	20	1 166.34	20	1 163.27
V_200_8	20	1 479.69	20	1 499.91	20	1 489.01	20	1 491.97	20	1 486.38
V_200_9	20	1 624.73	20	1 636.77	20	1 652.39	20	1 623.12	20	1 623.93
V_200_10	20	1 387.98	20	1 397.51	20	1 411.97	20	1 390.87	20	1 381.51
UBC_200_1	8	586.28	8	570.26	8	573.34	8	572.46	8	590.56
UBC_200_2	8	516.30	8	491.97	8	508.50	8	491.52	8	493.49
UBC_200_3	8	808.03	8	781.20	8	801.10	8	783.61	8	802.84
UBC_200_4	8	628.91	8	647.94	8	669.28	8	639.34	8	647.17
UBC_200_5	8	639.55	8	616.84	8	619.18	8	620.45	8	633.66
UBC_200_6	8	637.66	8	620.04	8	644.27	8	620.75	8	627.20
UBC_200_7	8	616.77	8	589.95	8	592.14	8	590.60	8	607.54
UBC_200_8	8	637.51	8	613.41	8	620.69	8	608.58	8	642.49
UBC_200_9	8	538.78	8	525.49	8	559.02	8	537.65	8	525.66
UBC_200_10	8	650.15	8	605.71	8	640.04	8	607.07	8	661.72

Table A.2 – Detailed results on the VRPDO instances with 200 customers in 2000 seconds

Instance	LNS		SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS	
	Route	Cost	Route	Cost	Route	Cost	Route	Cost	Route	Cost
U_400_1	41	1 895.00	41	1 943.08	41	1 971.02	41	1 877.14	41	1 920.64
U_400_2	41	3 182.56	41	3 125.90	41	3 242.58	41	3 067.01	41	3 136.19
U_400_3	41	2 240.91	41	2 245.68	41	2 294.83	41	2 205.32	41	2 251.91
U_400_4	40	1 827.88	40	1 822.91	40	1 867.62	40	1 783.72	40	1 801.16
U_400_5	40	2 219.25	40	2 256.24	40	2 277.34	40	2 218.57	40	2 227.34
U_400_6	40	2 006.75	40	2 001.19	40	2 084.98	40	1 967.86	40	1 993.56
U_400_7	41	2 608.05	41	2 545.98	41	2 605.71	41	2 483.67	41	2 549.34
U_400_8	41	1 949.08	41	1 972.34	41	1 994.32	41	1 940.24	41	1 976.39
U_400_9	40	2 703.21	40	2 682.11	40	2 715.64	40	2 645.95	40	2 768.49
U_400_10	41	2 188.59	41	2 241.12	41	2 280.78	41	2 169.11	41	2 231.91
V_400_1	41	2 504.80	41	2 517.53	41	2 556.16	41	2 499.38	41	2 516.62
V_400_2	41	2 728.18	41	2 725.34	41	2 729.75	41	2 709.91	41	2 719.09
V_400_3	40	1 898.97	40	1 884.05	40	1 939.59	40	1 862.73	40	1 862.73
V_400_4	40	1 919.80	40	1 924.06	40	1 962.68	40	1 891.02	40	1 903.72
V_400_5	41	1 743.08	41	1 748.98	41	1 776.06	41	1 731.56	41	1 746.66
V_400_6	40	2 824.44	40	2 777.59	40	2 820.20	40	2 745.29	40	2 772.18
V_400_7	41	2 066.69	41	2 074.31	41	2 098.70	41	2 067.62	41	2 051.71
V_400_8	41	1 901.45	41	1 934.74	41	1 948.57	41	1 881.84	41	1 908.00
V_400_9	40	2 303.40	40	2 322.51	40	2 359.74	40	2 298.65	40	2 324.23
V_400_10	40	2 665.05	40	2 645.68	40	2 692.97	40	2 654.08	40	2 665.31
UBC_400_1	16	1 042.41	16	976.29	16	1 017.37	16	1 001.78	16	1 009.93
UBC_400_2	16	1 226.77	16	1 149.60	16	1 194.05	16	1 141.70	16	1 163.04
UBC_400_3	15	1 430.09	15	1 160.43	15	1 215.95	15	1 122.51	15	1 158.75
UBC_400_4	16	1 264.99	16	1 153.68	16	1 198.04	16	1 162.55	16	1 193.29
UBC_400_5	16	1 077.65	16	1 004.50	16	1 064.95	16	1 035.19	16	1 058.48
UBC_400_6	15	1 576.22	15	1 283.89	15	1 338.55	15	1 283.39	15	1 329.49
UBC_400_7	15	1 104.06	15	981.88	15	1 046.79	15	987.64	15	1 026.01
UBC_400_8	16	1 280.05	16	1 199.68	16	1 243.71	16	1 188.97	16	1 231.85
UBC_400_9	16	1 243.39	16	1 148.06	16	1 154.68	16	1 142.94	16	1 162.20
UBC_400_10	15	1 550.53	15	1 165.35	15	1 282.90	15	1 221.59	15	1 296.37

Table A.3 – Detailed results on the VRPDO instances with 400 customers in 6000 seconds

A.2 Detailed results on the VRPTW instances

Instance	LNS		SLNS		Instance	LNS		SLNS		Instance	LNS		SLNS	
	Route	Cost	Route	Cost		Route	Cost	Route	Cost		Route	Cost	Route	Cost
c101	10	828.937	10	828.937	r101	19	1 657.180	19	1 650.800	rc101	14	1 696.950	14	1 696.950
c102	10	828.937	10	828.937	r102	17	1 490.250	17	1 486.120	rc102	12	1 554.750	12	1 554.750
c103	10	828.065	10	828.065	r103	13	1 315.230	14	1 213.620	rc103	11	1 268.360	11	1 262.020
c104	10	824.777	10	824.777	r104	9	1 032.080	9	1 007.310	rc104	10	1 138.410	10	1 135.830
c105	10	828.937	10	828.937	r105	14	1 377.110	14	1 377.110	rc105	14	1 542.460	14	1 540.180
c106	10	828.937	10	828.937	r106	12	1 253.210	12	1 252.030	rc106	11	1 424.730	12	1 376.260
c107	10	828.937	10	828.937	r107	10	1 113.010	10	1 112.370	rc107	11	1 232.120	11	1 230.950
c108	10	828.937	10	828.937	r108	9	972.969	9	964.472	rc108	10	1 140.360	10	1 139.820
c109	10	828.937	10	828.937	r109	11	1 204.250	11	1 197.420	rc201	4	1 413.520	4	1 423.510
c201	3	591.557	3	591.557	r110	10	1 142.560	10	1 151.680	rc202	3	1 419.240	3	1 389.090
c202	3	591.557	3	591.557	r111	10	1 099.300	10	1 096.740	rc203	3	1 052.350	3	1 064.140
c203	3	591.173	3	591.173	r112	9	1 006.470	9	998.858	rc204	3	799.061	3	798.464
c204	3	590.599	3	590.599	r201	4	1 253.230	4	1 252.370	rc205	4	1 302.930	4	1 302.420
c205	3	588.876	3	588.876	r202	3	1 196.620	3	1 195.300	rc206	3	1 146.320	3	1 161.280
c206	3	588.493	3	588.493	r203	3	942.978	3	946.624	rc207	3	1 086.160	3	1 081.230
c207	3	588.286	3	588.286	r204	2	842.440	2	837.950	rc208	3	828.709	3	834.077
c208	3	588.324	3	588.324	r205	3	994.428	3	994.428					
					r206	3	907.337	3	913.682					
					r207	2	904.432	2	905.440					
					r208	2	726.823	2	727.687					
					r209	3	914.342	3	917.132					
					r210	3	955.341	3	942.106					
					r211	2	891.115	2	898.839					

Table A.4 – Results on the VRPTW instances with 100 customers of Solomon and Desrosiers (1988)

Instance	LNS		SLNS		Instance	LNS		SLNS		Instance	LNS		SLNS	
	Route	Cost	Route	Cost		Route	Cost	Route	Cost		Route	Cost	Route	Cost
C1-2-1	20	2 704.570	20	2 704.570	R1-2-1	20	4 930.290	20	4 803.860	RC1-2-1	18	3 745.840	18	3 681.470
C1-2-2	18	2 944.350	18	2 960.380	R1-2-2	18	4 210.250	18	4 104.080	RC1-2-2	18	3 386.070	18	3 276.270
C1-2-3	18	2 761.630	18	2 720.570	R1-2-3	18	3 560.410	18	3 431.850	RC1-2-3	18	3 175.820	18	3 084.070
C1-2-4	18	2 715.100	18	2 643.310	R1-2-4	18	3 254.720	18	3 068.710	RC1-2-4	18	3 020.800	18	2 868.030
C1-2-5	20	2 702.050	20	2 702.050	R1-2-5	18	4 274.960	18	4 180.840	RC1-2-5	18	3 514.090	18	3 426.660
C1-2-6	20	2 701.040	20	2 701.040	R1-2-6	18	3 798.920	18	3 653.530	RC1-2-6	18	3 485.920	18	3 414.240
C1-2-7	20	2 701.040	20	2 701.040	R1-2-7	18	3 327.550	18	3 238.870	RC1-2-7	18	3 298.840	18	3 206.370
C1-2-8	19	2 783.260	19	2 775.480	R1-2-8	18	3 129.640	18	2 968.910	RC1-2-8	18	3 287.400	18	3 140.410
C1-2-9	18	2 745.530	18	2 687.830	R1-2-9	18	3 942.540	18	3 817.750	RC1-2-9	18	3 278.170	18	3 098.680
C1-2-10	18	2 723.810	18	2 643.550	R1-2-10	18	3 459.170	18	3 353.130	RC1-2-10	18	3 193.990	18	3 019.240
C2-2-1	6	1 931.440	6	1 931.440	R2-2-1	4	4 574.000	4	4 550.230	RC2-2-1	6	3 147.340	6	3 131.380
C2-2-2	6	1 863.160	6	1 863.160	R2-2-2	4	3 704.770	4	3 657.540	RC2-2-2	5	2 848.820	5	2 869.580
C2-2-3	6	1 779.200	6	1 775.080	R2-2-3	4	2 945.690	4	2 933.710	RC2-2-3	4	2 675.000	4	2 652.240
C2-2-4	6	1 715.990	6	1 703.430	R2-2-4	4	1 986.510	4	1 981.300	RC2-2-4	4	2 087.180	4	2 093.570
C2-2-5	6	1 878.850	6	1 878.850	R2-2-5	4	3 380.950	4	3 439.170	RC2-2-5	4	2 926.200	4	2 942.120
C2-2-6	6	1 857.830	6	1 857.350	R2-2-6	4	2 926.880	4	2 947.940	RC2-2-6	4	3 007.900	4	3 040.370
C2-2-7	6	1 849.460	6	1 849.460	R2-2-7	4	2 470.610	4	2 454.290	RC2-2-7	4	2 556.900	4	2 556.840
C2-2-8	6	1 826.210	6	1 820.530	R2-2-8	4	1 850.950	4	1 849.870	RC2-2-8	4	2 331.490	4	2 316.380
C2-2-9	6	1 832.590	6	1 830.050	R2-2-9	4	3 112.010	4	3 120.260	RC2-2-9	4	2 201.030	4	2 192.460
C2-2-10	6	1 808.210	6	1 808.210	R2-2-10	4	2 668.400	4	2 666.100	RC2-2-10	4	2 018.530	4	2 022.950

Table A.5 – Results on the VRPTW instances with 200 customers of Gehring and Homberger (1999)

Instance	LNS		SLNS		Instance	LNS		SLNS		Instance	LNS		SLNS	
	Route	Cost	Route	Cost		Route	Cost	Route	Cost		Route	Cost	Route	Cost
C1-4-1	40	7 152.060	40	7 152.060	R1-4-1	40	11 399.200	40	10 507.700	RC1-4-1	36	9 677.710	37	8 695.190
C1-4-2	36	9 190.420	37	7 337.870	R1-4-2	36	10 969.100	36	9 255.950	RC1-4-2	36	9 328.230	36	8 195.670
C1-4-3	36	8 214.670	36	7 170.080	R1-4-3	36	9 655.580	36	7 965.040	RC1-4-3	36	9 251.250	36	7 728.120
C1-4-4	36	8 328.280	36	6 848.330	R1-4-4	36	8 929.910	36	7 441.770	RC1-4-4	36	8 882.520	36	7 445.720
C1-4-5	40	7 164.680	40	7 152.060	R1-4-5	36	11 169.400	36	9 490.850	RC1-4-5	36	9 419.890	36	8 365.380
C1-4-6	40	7 154.620	40	7 153.450	R1-4-6	36	10 514.500	36	8 585.210	RC1-4-6	36	9 471.730	36	8 391.220
C1-4-7	39	7 622.700	39	7 561.490	R1-4-7	36	9 337.540	36	7 746.990	RC1-4-7	36	9 405.240	36	8 218.630
C1-4-8	37	8 065.370	37	7 680.290	R1-4-8	36	8 846.730	36	7 370.790	RC1-4-8	36	9 270.340	36	7 990.170
C1-4-9	36	8 765.160	36	7 320.370	R1-4-9	36	10 908.600	36	8 999.640	RC1-4-9	36	9 290.340	36	7 926.390
C1-4-10	36	8 219.410	36	6 990.770	R1-4-10	36	9 922.990	36	8 348.890	RC1-4-10	36	9 181.690	36	7 771.380
C2-4-1	12	4 126.560	12	4 116.140	R2-4-1	8	9 416.910	8	9 423.990	RC2-4-1	11	6 950.180	11	6 890.840
C2-4-2	12	4 002.360	12	3 963.930	R2-4-2	8	7 827.640	8	7 760.080	RC2-4-2	10	6 059.030	10	6 063.870
C2-4-3	12	3 880.370	12	3 807.490	R2-4-3	8	6 098.400	8	6 144.680	RC2-4-3	8	5 133.490	8	5 154.130
C2-4-4	11	4 347.170	11	3 965.950	R2-4-4	8	4 372.600	8	4 378.920	RC2-4-4	8	3 739.760	8	3 750.300
C2-4-5	12	3 977.900	12	4 084.320	R2-4-5	8	7 365.950	8	7 300.280	RC2-4-5	9	6 275.300	9	6 207.720
C2-4-6	12	3 908.970	12	3 917.060	R2-4-6	8	6 292.330	8	6 299.700	RC2-4-6	8	5 999.660	8	6 071.860
C2-4-7	12	3 940.830	12	3 933.210	R2-4-7	8	5 145.110	8	5 172.360	RC2-4-7	8	5 525.090	8	5 499.460
C2-4-8	12	3 875.500	12	3 818.870	R2-4-8	8	4 121.390	8	4 096.350	RC2-4-8	8	4 950.320	8	4 895.760
C2-4-9	12	3 936.450	12	3 884.330	R2-4-9	8	6 570.770	8	6 556.620	RC2-4-9	8	4 655.850	8	4 665.900
C2-4-10	11	4 323.370	12	3 733.090	R2-4-10	8	5 966.820	8	5 949.860	RC2-4-10	8	4 414.350	8	4 341.230

Table A.6 – Results on the VRPTW instances with 400 customers of Gehring and Homberger (1999)

A.3 Detailed results on the GVRPTW instances

Instance	SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS		
	Route	Cost	Route	Cost	Route	Cost	Route	Cost	Time (s)
i-030-04-08	4	3 497	4	3 497	4	3 497	4	3 497	5
i-030-08-12	4	2 796	4	2 796	4	2 796	4	2 796	5
i-040-04-08	6	3 811	6	3 811	6	3 811	6	3 811	8
i-040-08-12	6	3 768	6	3 768	6	3 768	6	3 768	8
i-050-04-08	8	5 447	8	5 447	8	5 439	8	5 447	11
i-050-08-12	8	4 034	8	4 034	8	4 034	8	4 034	11
i-060-04-08	8	5 908	8	5 926	8	5 919	8	5 926	14
i-060-08-12	8	4 303	8	4 303	8	4 303	8	4 332	14
i-070-04-08	10	6 246	10	6 246	10	6 224	10	6 246	22
i-070-08-12	10	4 644	10	4 644	10	4 644	10	4 644	22
i-080-04-08	12	7 390	12	7 394	12	7 394	12	7 394	27
i-080-08-12	12	5 686	12	5 661	12	5 692	12	5 661	27
i-090-04-08	11	7 187	11	7 182	11	7 187	11	7 215	32
i-090-08-12	11	5 903	11	5 869	11	5 830	11	5 808	32
i-100-04-08	14	7 308	14	7 349	14	7 295	14	7 295	38
i-100-08-12	14	6 546	14	6 546	14	6 585	14	6 606	38
i-110-04-08	16	8 696	16	8 720	16	8 711	16	8 687	45
i-110-08-12	16	6 249	16	6 487	16	6 338	16	6 310	45
i-120-04-08	15	8 344	15	8 357	15	8 344	15	8 344	60
i-120-08-12	15	6 829	15	6 829	15	6 774	15	6 774	60

Table A.7 – Detailed results on the GVRPTW instances of Moccia et al. (2012) with their number of vehicles

A.4 Detailed results on the VRPRDL and VRPHRDL instances

Instance	SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS		
	Route	Cost	Route	Cost	Route	Cost	Route	Cost	Time (s)
instance_0	10	3 203	10	3 203	10	3 203	10	3 203	10
instance_1	9	2 799	9	2 799	9	2 799	9	2 799	10
instance_2	8	2 603	8	2 607	8	2 603	8	2 603	10
instance_3	7	2 261	7	2 261	7	2 261	7	2 261	10
instance_4	10	3 217	10	3 217	10	3 217	10	3 217	10
instance_5	9	2 805	9	2 805	9	2 805	9	2 805	10
instance_6	10	3 339	10	3 339	10	3 339	10	3 339	10
instance_7	10	3 325	10	3 325	10	3 325	10	3 325	10
instance_8	11	3 534	11	3 534	11	3 534	11	3 534	10
instance_9	10	2 752	10	2 752	10	2 752	10	2 752	10
variant2_instance_0	7	2 133	7	2 133	7	2 133	7	2 133	10
variant2_instance_1	6	1 946	6	1 946	6	1 946	6	1 946	10
variant2_instance_2	8	1 966	8	1 966	8	1 966	8	1 966	10
variant2_instance_3	6	1 610	6	1 610	6	1 610	6	1 610	10
variant2_instance_4	8	2 478	8	2 478	8	2 478	8	2 478	10
variant2_instance_5	8	2 469	8	2 469	8	2 469	8	2 469	10
variant2_instance_6	7	1 946	7	1 946	7	1 946	7	1 946	10
variant2_instance_7	8	2 380	8	2 380	8	2 380	8	2 380	10
variant2_instance_8	8	2 492	8	2 492	8	2 492	8	2 492	10
variant2_instance_9	8	2 443	8	2 443	8	2 443	8	2 443	10

Table A.8 – Detailed results on the VRPRDL instances of Ozbaygin et al. (2017) with the number of vehicles of Tilk et al. (2020)

Instance	SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS		
	Route	Cost	Route	Cost	Route	Cost	Route	Cost	Time (s)
instance_0-triangle	4	901	4	901	4	901	4	901	3
instance_1-triangle	5	1 286	5	1 286	5	1 286	5	1 286	3
instance_2-triangle	4	991	4	991	4	991	4	991	3
instance_3-triangle	5	1 062	5	1 062	5	1 062	5	1 062	3
instance_4-triangle	6	1 832	6	1 832	6	1 832	6	1 832	3
instance_5-triangle	5	1 294	5	1 294	5	1 294	5	1 294	4
instance_6-triangle	4	1 155	4	1 155	4	1 155	4	1 155	4
instance_7-triangle	6	1 455	6	1 455	6	1 455	6	1 455	4
instance_8-triangle	5	1 260	5	1 260	5	1 260	5	1 260	4
instance_9-triangle	7	1 684	7	1 684	7	1 684	7	1 684	4
instance_10-triangle	7	1 922	7	1 922	7	1 922	7	1 922	8
instance_11-triangle	8	2 324	8	2 324	8	2 324	8	2 324	8
instance_12-triangle	6	1 747	6	1 747	6	1 747	6	1 747	8
instance_13-triangle	6	1 273	6	1 273	6	1 273	6	1 273	8
instance_14-triangle	6	1 694	6	1 694	6	1 694	6	1 694	8
instance_15-triangle	7	1 938	7	1 938	7	1 938	7	1 938	8
instance_16-triangle	8	1 965	8	1 965	8	1 965	8	1 965	8
instance_17-triangle	7	1 827	7	1 827	7	1 827	7	1 827	8
instance_18-triangle	7	2 083	7	2 083	7	2 083	7	2 083	8
instance_19-triangle	6	1 822	6	1 822	6	1 822	6	1 822	8
instance_20-triangle	13	3 761	13	3 761	13	3 761	13	3 761	12
instance_21-triangle	10	2 828	10	2 828	10	2 828	10	2 828	12
instance_22-triangle	16	4 440	16	4 440	16	4 440	16	4 440	12
instance_23-triangle	11	3 378	11	3 378	11	3 378	11	3 378	12
instance_24-triangle	11	3 161	11	3 161	11	3 161	11	3 161	12
instance_25-triangle	16	4 536	16	4 536	16	4 536	16	4 536	12
instance_26-triangle	10	2 865	10	2 865	10	2 865	10	2 865	12
instance_27-triangle	14	4 173	14	4 173	14	4 173	14	4 173	12
instance_28-triangle	14	3 964	14	3 964	14	3 964	14	3 964	12
instance_29-triangle	14	4 107	14	4 107	14	4 107	14	4 107	12
instance_30-triangle	17	4 935	17	4 935	17	4 935	17	4 935	60
instance_31-triangle	18	5 258	18	5 267	18	5 258	18	5 258	60
instance_32-triangle	18	5 061	18	5 061	18	5 061	18	5 061	60
instance_33-triangle	17	5 218	17	5 220	17	5 218	17	5 218	60
instance_34-triangle	20	5 498	20	5 498	20	5 498	20	5 521	60
instance_35-triangle	22	6 498	22	6 621	22	6 498	22	6 498	60
instance_36-triangle	17	4 830	17	4 830	17	4 830	17	4 830	60
instance_37-triangle	22	5 604	22	5 604	22	5 604	22	5 604	60
instance_38-triangle	21	5 841	21	5 841	21	5 841	21	5 841	60
instance_39-triangle	17	5 016	17	4 999	17	4 995	17	4 995	60

Table A.9 – Detailed results on the VRPRDL instances of Reyes et al. (2017) with the number of vehicles of Tilk et al. (2020)

Instance	SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS		
	Route	Cost	Route	Cost	Route	Cost	Route	Cost	Time (s)
instance_0-triangle	3	773	3	773	3	773	3	773	3
instance_1-triangle	4	1 065	4	1 065	4	1 065	4	1 065	3
instance_2-triangle	3	988	3	988	3	988	3	988	3
instance_3-triangle	3	914	3	914	3	914	3	914	3
instance_4-triangle	6	1 710	6	1 710	6	1 710	6	1 710	3
instance_5-triangle	4	1 099	4	1 099	4	1 099	4	1 099	4
instance_6-triangle	3	996	3	996	3	996	3	996	4
instance_7-triangle	5	1 346	5	1 346	5	1 346	5	1 346	4
instance_8-triangle	4	997	4	997	4	997	4	997	4
instance_9-triangle	4	1 166	4	1 166	4	1 166	4	1 166	4
instance_10-triangle	5	1 596	5	1 596	5	1 596	5	1 596	8
instance_11-triangle	6	1 808	6	1 808	6	1 808	6	1 808	8
instance_12-triangle	6	1 563	6	1 563	6	1 563	6	1 563	8
instance_13-triangle	4	1 058	4	1 058	4	1 058	4	1 058	8
instance_14-triangle	5	1 347	5	1 347	5	1 347	5	1 347	8
instance_15-triangle	5	1 517	5	1 517	5	1 517	5	1 517	8
instance_16-triangle	5	1 445	5	1 445	5	1 445	5	1 445	8
instance_17-triangle	5	1 627	5	1 627	5	1 627	5	1 627	8
instance_18-triangle	5	1 461	5	1 461	5	1 461	5	1 461	8
instance_19-triangle	6	1 715	6	1 715	6	1 715	6	1 715	8
instance_20-triangle	8	2 598	8	2 586	8	2 580	8	2 580	12
instance_21-triangle	7	2 206	7	2 206	7	2 206	7	2 206	12
instance_22-triangle	10	3 363	10	3 363	10	3 363	10	3 363	12
instance_23-triangle	8	2 569	8	2 569	8	2 569	8	2 569	12
instance_24-triangle	8	2 384	8	2 383	8	2 381	8	2 389	12
instance_25-triangle	9	2 845	9	2 845	9	2 845	9	2 845	12
instance_26-triangle	8	2 518	8	2 518	8	2 518	8	2 518	12
instance_27-triangle	8	2 758	8	2 758	8	2 758	8	2 758	12
instance_28-triangle	9	2 892	9	2 892	9	2 892	9	2 892	12
instance_29-triangle	8	2 691	8	2 691	8	2 691	8	2 691	12
instance_30-triangle	12	3 666	12	3 765	12	3 666	12	3 726	60
instance_31-triangle	13	3 885	13	3 885	13	3 885	13	3 885	60
instance_32-triangle	13	3 554	13	3 588	13	3 543	13	3 557	60
instance_33-triangle	12	3 767	12	3 783	12.2	3 784	12	3 751	60
instance_34-triangle	11	3 184	11	3 185	11	3 184	11	3 175	60
instance_35-triangle	15	4 287	15	4 379	15	4 273	15	4 303	60
instance_36-triangle	10	3 225	10.5	3 370	10	3 217	10	3 217	60
instance_37-triangle	14	3 935	14	3 939	14	3 936	14	3 937	60
instance_38-triangle	15	4 300	15	4 335	15	4 300	15	4 300	60
instance_39-triangle	13	3 556	13	3 557	13	3 550	13	3 556	60

Table A.10 – Detailed results on the VRPHRDL instances of Reyes et al. (2017) with the number of vehicles of Tilk et al. (2020)

Instance	SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS		
	Route	Cost	Route	Cost	Route	Cost	Route	Cost	Time (s)
instance_0	8	2 662	8	2 662	8	2 662	8	2 662	10
instance_1	8	2 610	8	2 610	8	2 610	8	2 610	10
instance_2	7	2 260	7	2 260	7	2 260	7	2 260	10
instance_3	7	2 147	7	2 147	7	2 147	7	2 147	10
instance_4	10	3 172	10	3 172	10	3 172	10	3 172	10
instance_5	8	2 616	8	2 616	8	2 616	8	2 616	10
instance_6	9	3 010	9	3 010	9	3 010	9	3 010	10
instance_7	10	3 278	10	3 278	10	3 278	10	3 278	10
instance_8	11	3 514	11	3 514	11	3 514	11	3 514	10
instance_9	9	2 727	9	2 727	9	2 727	9	2 727	10
variant2_instance_0	6.6	2 070	6.6	2 109	6.2	2 019	6.8	2 096	10
variant2_instance_1	6	1 931	6	1 931	6	1 931	6	1 930	10
variant2_instance_2	6	1 830	6	1 830	6	1 830	6	1 830	10
variant2_instance_3	5	1 478	5	1 478	5	1 478	5	1 478	10
variant2_instance_4	8	2 466	8	2 466	8	2 466	8	2 466	10
variant2_instance_5	8	2 388	8	2 388	8	2 388	8	2 388	10
variant2_instance_6	6	1 848	6	1 854	6	1 854	6	1 861	10
variant2_instance_7	7	2 264	7	2 266	7	2 265	7	2 266	10
variant2_instance_8	8	2 457	8	2 457	8	2 457	8	2 457	10
variant2_instance_9	7.5	2 419	7.5	2 364	7.5	2 330	7	2 302	10

Table A.11 – Detailed results on the VRPHRDL instances of Ozbaygin et al. (2017) with the number of vehicles of Tilk et al. (2020)

Instance	VRPRDL		VRPHRDL		
	Route	Cost	Route	Cost	Time (s)
instance_0-triangle	4	901	3	773	3
instance_1-triangle	5	1286	4	1065	3
instance_2-triangle	4	991	3	988	3
instance_3-triangle	5	1062	3	914	3
instance_4-triangle	6	1832	6	1710	3
instance_5-triangle	5	1294	4	1099	4
instance_6-triangle	4	1155	3	996	4
instance_7-triangle	6	1455	5	1346	4
instance_8-triangle	5	1260	4	997	4
instance_9-triangle	8	1684	4	1166	4
instance_10-triangle	7	1922	5	1587	5
instance_11-triangle	8	2324	6	1808	5
instance_12-triangle	6	1747	6	1563	5
instance_13-triangle	6	1273	4	1058	5
instance_14-triangle	6	1694	5	1347	5
instance_15-triangle	7	1938	5	1517	5
instance_16-triangle	8	1965	5	1445	5
instance_17-triangle	7	1827	5	1627	5
instance_18-triangle	7	2083	5	1461	5
instance_19-triangle	6	1822	6	1715	7
instance_20-triangle	13	3761	8	2580	7
instance_21-triangle	10	2828	7	2206	7
instance_22-triangle	16	4440	10	3363	7
instance_23-triangle	11	3378	8	2569	7
instance_24-triangle	11	3161	8	2378	7
instance_25-triangle	16	4536	9	2845	7
instance_26-triangle	10	2865	8	2518	7
instance_27-triangle	14	4173	8	2758	7
instance_28-triangle	14	3964	9	2892	7
instance_29-triangle	14	4107	8	2691	35
instance_30-triangle	18	4935	14	3666	35
instance_31-triangle	19	5258	13	3885	35
instance_32-triangle	18	5061	13	3543	35
instance_33-triangle	17	5218	13	3694	35
instance_34-triangle	20	5498	11	3184	35
instance_35-triangle	22	6498	15	4273	35
instance_36-triangle	17	4830	11	3217	35
instance_37-triangle	21	5604	14	3935	35
instance_38-triangle	24	5841	15	4300	35
instance_39-triangle	19	4995	13	3537	35

Table A.12 – Detailed results of SLNS+SPP on the VRPRDL and VRPHRDL instances of Reyes et al. (2017) with the number of vehicles of Ozbaygin et al. (2017)

Instance	VRPRDL		VRPHRDL		
	Route	Cost	Route	Cost	Time (s)
instance_0	10	3203	8	2662	6
instance_1	9	2799	8	2610	6
instance_2	8	2603	7	2260	6
instance_3	7	2261	7	2147	6
instance_4	10	3217	10	3172	6
instance_5	9	2805	8	2616	6
instance_6	10	3339	9	3010	6
instance_7	10	3325	10	3278	6
instance_8	11	3534	11	3514	6
instance_9	10	2752	10	2727	6
variant2_instance_0	7	2133	6	1998	6
variant2_instance_1	7	1946	6	1927	6
variant2_instance_2	8	1966	6	1830	6
variant2_instance_3	6	1610	5	1478	6
variant2_instance_4	8	2478	8	2466	6
variant2_instance_5	8	2469	8	2388	6
variant2_instance_6	7	1946	6	1848	6
variant2_instance_7	8	2380	7	2264	6
variant2_instance_8	8	2492	8	2457	6
variant2_instance_9	8	2443	7	2302	6

Table A.13 – Detailed results of SLNS+SPP on the VRPRDL and VRPHRDL instances of Ozbaygin et al. (2017) with the number of vehicles of Ozbaygin et al. (2017)

A.5 Detailed results on the VRPMTW instances

Instance	SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS	
	Route	Cost	Route	Cost	Route	Cost	Route	Cost
rm101	10	972.31	10	969.68	10	967.83	10	967.83
rm102	9	930.81	9	915.26	9	903.94	9	911.56
rm103	9	882.51	9	882.51	9	882.51	9	882.51
rm104	9	888.28	9	888.28	9	888.28	9	888.28
rm105	9	882.04	9	882.04	9	882.04	9	882.04
rm106	9	899.98	9	899.98	9	899.98	9	899.98
rm107	9	879.37	9	879.37	9	879.37	9	879.37
rm108	9	913.71	9	913.71	9	913.71	9	913.71
rm201	2	756.19	2	752.93	2	761.66	2	758.98
rm202	2	682.56	2	683.55	2	685.61	2	687.11
rm203	2	679.19	2	679.20	2	679.20	2	679.67
rm204	2	672.83	2	672.83	2	672.83	2	672.83
rm205	2	671.26	2	671.02	2	671.02	2	671.35
rm206	2	678.96	2	678.96	2	678.96	2	678.96
rm207	2	674.39	2	674.39	2	674.39	2	674.73
rm208	2	673.93	2	673.93	2	676.73	2	675.71
cm101	10	1 112.17	10	1 120.81	10	1 091.12	10	1 060.20
cm102	12	1 045.34	11	1 170.01	11	1 126.31	11	1 105.40
cm103	11	1 216.32	11	1 220.73	11	1 215.32	11	1 229.29
cm104	13	1 259.63	13	1 273.44	13	1 259.63	13	1 259.63
cm105	10	999.28	10	999.28	10	999.28	10	999.28
cm106	9	1 175.63	9	1 208.55	9	1 176.08	9	1 153.11
cm107	10	1 077.08	10	1 077.08	10	1 077.08	10	1 077.08
cm108	9	1 068.82	9	1 080.61	9	1 064.46	9	1 066.65

Table A.14 – Detailed results on the VRPMTW instances of Belhaiza et al. (2014) in 600 seconds (part 1)

Instance	SLNS		SLNS+BS		SLNS+SPP		SLNS+SPP+BS	
	Route	Cost	Route	Cost	Route	Cost	Route	Cost
cm201	5	865.56	5	895.87	5	876.06	5	871.14
cm202	6	782.41	6	791.36	6	782.41	6	782.41
cm203	5	938.63	5	952.90	5	946.79	5	932.78
cm204	5	817.48	5	817.48	5	817.48	5	817.48
cm205	4	1 032.08	4	1 022.36	4	1 020.57	4	1 000.65
cm206	4	895.99	4	899.93	4	895.99	4	895.99
cm207	4	1 119.35	4	1 125.86	4	1 125.86	4	1 130.66
cm208	4	913.88	4	913.88	4	913.88	4	913.88
rcm101	10	1 074.01	10	1 074.04	10	1 074.01	10	1 074.01
rcm102	10	1 121.62	10	1 121.62	10	1 121.62	10	1 121.62
rcm103	10	1 120.89	10	1 120.89	10	1 120.89	10	1 120.89
rcm104	10	1 124.29	10	1 124.29	10	1 124.29	10	1 124.29
rcm105	10	1 165.35	10	1 165.35	10	1 165.35	10	1 165.35
rcm106	10	1 165.74	10	1 167.73	10	1 165.74	10	1 165.74
rcm107	11	1 290.83	11	1 296.36	11	1 290.83	11	1 295.38
rcm108	11	1 342.45	11	1 345.45	11	1 339.00	11	1 339.00
rcm201	2	716.49	2	735.76	2	698.33	2	727.35
rcm202	2	719.48	2	727.25	2	732.48	2	722.86
rcm203	2	704.76	2	704.76	2	710.67	2	704.76
rcm204	2	692.62	2	692.21	2	692.62	2	692.21
rcm205	2	711.63	2	718.93	2	723.65	2	718.77
rcm206	2	732.36	2	721.86	2	736.89	2	721.86
rcm207	2	864.97	2	857.63	2	863.73	2	867.93
rcm208	2	722.68	2	722.68	2	722.68	2	722.68

Table A.15 – Detailed results on the VRPMTW instances of Belhaiza et al. (2014) in 600 seconds (part 2)

DETAILS ON ITERATIVE TWO-STAGES HEURISTIC

B.1 Details about the SLNS for the 2E-MTVRP-CSRF used in ITSH

In this section, we detail the Large Neighborhood Search heuristics LNS[1] and LNS[2]. They are both based on Algorithm 5, but present slight differences in the LNS operators and parameters.

B.1.1 LNS Algorithm

The algorithm starts from an initial solution x, obtained as described in Section B.1.5. x^* denotes the best solution found ; it is returned at the end of the algorithm. The pool H stores the trips of the considered echelon. The index i counts the number of iterations performed since the best-found solution x^* was improved or since the last large destruction happened. We denote by Σ^+ the set of repair operators and by Σ^- the set of destroy operators. $\Sigma^-|_{\text{local}}$ denotes the set of local destroy operators, i.e operators used for small destructions. The operators used for each echelon are presented later.

Line (6) shows the LNS stopping criterion. We call a *cycle* a sequence of iterations with small destruction followed by an iteration with large destruction. LNS stops after a predefined number ζ of such cycles. In addition, we also stop the algorithm if no feasible solution (serving all customers/FEDs) were found after ζ' cycles. We set $\zeta' << \zeta$ in order to speed-up the interactions between the two LNS to find a feasible solution during the earlier stages.

On each iteration, a repair operator is randomly selected in Σ^+ (line (7)). A small destruction is performed on a copy of the current solution (line (9)). Then a local destroy operator is selected in $\Sigma^-|_{\text{local}}$ and the destruction size is randomly selected in $[\delta_{\text{small}}, \Delta_{\text{small}}]$. The selected operators, σ^- and σ^+, are then applied to the solution x' at line (19).

A large destruction is performed if $i = \omega$ at line (8). This happens if the best-found solution has not been improved for ω successive iterations with small destruction. In this case, the destroy operator is randomly selected in Σ^- and the destruction size in $[\delta_{\text{large}}, \Delta_{\text{large}}]$, at lines (14) and (15) respectively. We assume that $\delta_{\text{large}} > \delta_{\text{small}}$ and that $\Delta_{\text{large}} > \Delta_{\text{small}}$. In addition, a large destruction starts from the best-found solution x^*. The new solution x' is then produced at line (19). Note that, after a large destruction, the produced solution is unreservedly accepted as the new current solution x on line (24).

At line (23), the trips of the newly generated solution are added to the pool of trips H.

Following a small destruction, the new solution x' is accepted as the new current solution x (line (27)) only if it is not $\epsilon\%$ worse than the best solution x^*. At lines (31)–(33), the value of x^* is updated and the counter i is set to the value 0 when a new best solution is found.

Algorithm 5: LNS with large and small destruction

1: x, x^* : initial solution
2: pool of trips $H \leftarrow \emptyset$
3: $i \leftarrow 0$
4: $cycle \leftarrow 0$
5: $i' \leftarrow 0$
6: **while** $cycle < \zeta$ **and** $i' < \zeta'$ **do**
7:     randomly select a repair operator $\sigma^+ \in \Sigma^+$
8:     **if** $i < \omega$ **then**
9:         $x' \leftarrow x$
10:         randomly select a destroy operator $\sigma^- \in \Sigma^-|_{\text{local}}$
11:         randomly select a destruction size $\Phi \in [\delta_{\text{small}}, \Delta_{\text{small}}]$
12:     **else**
13:         $x' \leftarrow x^*$
14:         randomly select a destroy operator $\sigma^- \in \Sigma^-$
15:         randomly select a destruction size $\Phi \in [\delta_{\text{large}}, \Delta_{\text{large}}]$
16:         $cycle \leftarrow cycle + 1$
17:         $i' \leftarrow i' + 1$
18:     **end if**
19:     $x' \leftarrow \sigma^+(\sigma^-(x', \Phi))$
20:     **if** x' is feasible **then**
21:         $i' \leftarrow 0$
22:     **end if**
23:     store the trips of x' into the pool H
24:     **if** $i = \omega$ **then**
25:         $x \leftarrow x'$
26:         $i \leftarrow 0$
27:     **else if** $f'(x') < (1 + \epsilon) \cdot f'(x^*)$ **then**
28:         $x \leftarrow x'$
29:         $i \leftarrow i + 1$
30:     **end if**
31:     **if** $f'(x') < f'(x^*)$ **then**
32:         $x^* \leftarrow x'$
33:         $i \leftarrow 0$
34:     **end if**
35: **end while**
36: **return** x^*

B.1.2 Operators

Below, we list all LNS operators used by the LNS algorithm. The symbols [1], [2] and [1,2] identify the operators used only to solve SP1 SP2or both subproblems, respectively. In the list below, the LNS operators listed with [1,2] are used to solve both SP1 and SP2, .

The **local destroy operators** are:

[1,2] *(Split) String removal* (Christiaens and Vanden Berghe, 2020): removes sequences of FEDs/customers in different routes of the considered solution. We also implement the *Split String Removal* version of this operator, which conserve a sub-string of vertices in the middle of considered string to be removed.

[1,2] *Satellite removal*: for each satellite, we call *satellite schedule* the list of vertices taking place at this satellite ordered in non-decreasing date $\mathcal{T}$. This operator removes sequences of operations in the satellite schedules of the considered solution, either conserving, or not, a sub-string of operations in the middle of the string to be removed. In SP2 if a FED is removed, the associate customer is removed from its second-echelon trip. This operator is equivalent to the (split) string removal on satellite schedules.

[2] *Distance related removal* (Ropke and Pisinger, 2006b): removes customers that are close to each other with respect to the Euclidean distance.

[2] *FEV removal*: randomly selects a first-echelon FEV to a satellite and removes it as-well-as all the FEDs assigned to this visit.

[2] *Cluster removal* (Pisinger and Ropke, 2007): removes customers that are served by the same trip in the considered solution. A trip is randomly selected and the Kruskal's algorithm is run on the arcs of this trip until two clusters remain. One cluster is randomly chosen and all its customers are removed from the solution.

The **large destroy operators** are:

[2] *Random removal* (Ropke and Pisinger, 2006b): randomly removes customers.

[1] *Random visit removal*: randomly selects FEVs at satellites and remove them. It is equivalent to random removal applied to FEVs.

[1] *Worst visit removal* (derived from worst removal, Ropke and Pisinger (2006a)): iteratively remove the FEV which removal decreases the objective function the most.

[2] *Historical knowledge node removal* (Demir et al., 2012): for this removal, the minimum cost of each customer is recorded over all past LNS iterations. The cost of a customer in a

solution x is the difference $f(x) - f(x')$, where x' is x without the service of the considered customer. The operator iteratively removes the customers with the largest difference between their current cost and their lowest cost. It can be seen as a history-biased worst removal.

[1,2] *Trip removal and Route removal* (Nagata and Bräysy, 2009): remove all the FEDs/customers and visits of a randomly selected trip/route.

Following Christiaens and Vanden Berghe (2020), the used **repair operators** are all lists heuristics. In these operators, unserved FEDs/customers are first sorted according to one of the following simple rules and then inserted one by one with the procedure described in Section 3.4.2. At each iteration, one rule is selected at random with the same selection probability for each rule.

[1,2] *Random order*: the FEDs/customers are not sorted before insertion.

[1,2] *Largest first*: FEDs/customers are sorted in non-increasing order of total demand quantity.

[2] *Farthest first*: customers are sorted in non-increasing order of distance to their nearest satellite.

[2] *Closest first*: customers are sorted in non-decreasing order of distance to their nearest satellite.

[1,2] *Earliest first*: FEDs/customers are sorted in non-decreasing order of ready-date or opening of time-window.

[1,2] *Latest first*: FEDs/customers are sorted in non-increasing order of due-date or time-window end.

[1,2] *Narrow first*: FEDs/customers are sorted in non-decreasing order of time-window width.

B.1.3   Acceptance criterion

We use the record-to-record acceptance criterion, we use the *modified cost* proposed in (Pisinger and Ropke, 2007):

$$f'(x) = z(x) \cdot \left(1 + \beta \cdot \frac{B(x)}{|N|}\right), \tag{B.1}$$

where $z(x)$ is the objective function as defined in Model 3, $B(x)$ is the number of unserved customers and FEDs, and β a penalty factor. In addition, in LNS[1], we replace $z(x)$ by $z'(x)$. This modified objective function is equal to $z(x)$ plus a penalty for successive visits of first-echelon

vehicles at the same satellite. This penalty is used to mitigate some unrealistic solutions in which first-echelon vehicles use some satellites as waiting stations where they have repeated visits.

B.1.4   LNS parameters

The numeric parameters for both LNS^1 and LNS^2 are summarized in Table B.1. The values of these parameters were obtained from preliminary experiments on 2E-MTVRP-CSRF and 2E-VRP instances.

Parameter	Notation	LNS^1	LNS^2
Large destruction frequency	ω	$9|N|^{0.75}$	$9|N|^{1.0}$
Number of large destructions	ζ	30	5
Number of large destructions without improvement	ζ'	5	1
Minimal size of a small destructions	δ_{small}	1%	1%
Maximal size of a small destructions	Δ_{small}	15%	10%
Minimal size of a large destructions	δ_{large}	20%	10%
Maximal size of a large destructions	Δ_{large}	100%	30%
Penalty for unserved FED/customer	β	10	25
Range of acceptance	ϵ	2%	2%

Table B.1 – LNS parameters

The other parameters are as follows:

— Penalty for successive visits at satellite by the first-echelon vehicles: 1.

— All the recreate operators are equiprobable in LNS^1 and LNS^2. All the local destroy operators are equiprobable in LNS^1 and LNS^2. All the large destroy operators are equiprobable in LNS^1 and LNS^2. During a large destruction, there is 20% chance of taking a local destroy operator.

B.1.5   Initial solution

During the first iteration of ITSH (Algorithm 4), SP2 is solved first by LNS^2(line 4). Thus, no routing of the first-echelon vehicles has been build yet. For each satellite, 4 single-trip routes are created. These trips are round trips from the depot to the satellite, evenly scheduled over the time horizon. This procedure is repeated F^1 times for each satellite. Thus, the initial dummy first-echelon solution uses far too many vehicles. In this way, the second-echelon can rely on nearly unlimited supply of the satellite by the first-echelon but still needs to consolidate its demand on given visits.

To initialize the second-echelon routes using the dummy first-echelon, each customer demand is assigned to its nearest satellite. Then, second-echelon vehicles are assigned to satellites such that the proportion of vehicles assigned to a satellite is proportional to the proportion of customers demands assigned to it. Each second-echelon vehicle route is initialized by: an empty trip from the second-level depot to its assigned satellite, an empty trip starting and ending at this satellite and finally, an empty trip from its satellite back to the depot. The customers are inserted in this empty second-echelon solution by the first call to a repair operator.

The supproblem SP1 is solved with LNS^1 for the first time at Line 6. The first-echelon routes are initialized with an empty trip from the depot to the depot. The first call to a repair operator initializes the trips.

During the subsequent iteration of Algorithm 4, the current solution (x^1, x^2) is passed on to be improved. The first-echelon routes x^1 will be used by LNS^2 and the second-echelon routes x^2 will be translated in FED in LNS^1. Each algorithm starting from the routes of x^2 and x^1 respectively.

B.2 Details about the modified model of the 2E-MTVRP-CSRF used in ITSH

The refined Model use the following additional variables:

$\sigma_i \geq 0$: Penalty-variable for not-serving customer i.

$u_{s,t}^{\text{fw}} \geq 0$: Surplus amount that is not dropped off at satellite s at time t by first-echelon trips.

$u_{s,t}^{\text{rv}} \geq 0$: Surplus amount that is not collected at satellite s at time t by first-echelon trips.

$\zeta_s \in \{0, 1\}$: State if satellite s is utilized or not.

Let $\bar{\Omega}^1$ be the set of the computed first-echelon trips and let $\bar{\Omega}^2$ be the set of all created second-echelon trips. We introduce the following sets of time periods to reduce the number of constraints:

1. $\mathbb{T}^{1,\text{st}} := \{t \in \mathbb{T} : \exists h = (P, T, L) \in \bar{\Omega}^1 \text{ with } s_0 = s \text{ and } T_0 = t\}$, the time periods at which a first-echelon vehicle starts from the CDC.

2. $\mathbb{T}_s^{1,\text{vis}} := \{t \in \mathbb{T} : \exists h = (P, T, L) \in \bar{\Omega}^1 \text{ with } s = i_k \in P \text{ and } T_{i_k} = t\}$, the time periods at which a first-echelon vehicle visits a satellite s.

3. $\mathbb{T}_s^{2,\text{st}} := \{t \in \mathbb{T} : H_{s,t}^{\bar{\Omega}^2+} \neq \emptyset\}$, the time periods at which a second-echelon vehicle starts at satellite s.

4. $\mathbb{T}_s^{2,\text{end}} := \{t \in \mathbb{T} : H_{s,t}^{\bar{\Omega}^2-} \neq \emptyset\}$, the time periods at which a second-echelon vehicle ends at satellite s.

Moreover, we remove constraints (3.b)–(3.g) for those time periods which are dominated. A $\leq$-constraint for a time period t is considered dominated if the $\leq$-constraint for time period $t+1$ is tighter, i.e., the left-hand side contains more positive and/or less negative terms. Similarly, a $\geq$-constraint is tighter if the left-hand side contains less positive and/or more negative terms. For example, in constraints (3.e) for satellite s, it suffices to consider only those points in time, at which a second-echelon vehicle starts a trip to deliver customers.

Model 4: Enhanced Formulation

$$min \sum_{h \in \bar{\Omega}^1} c_h^1 x_h^1 + \sum_{h \in \bar{\Omega}^2} c_h^2 x_h^2 + \sum_{i \in N} M \sigma_i$$

$$s.t. \sum_{h \in \bar{\Omega}^2} \alpha_{h,i} x_h^2 + \sigma_i = 1 \qquad \forall i \in N \qquad (4.a)$$

$$\sum_{h \in \bar{\Omega}^1} \beta_{h,t} x_h^1 \leq |F^1| \qquad \forall t \in \mathbb{T}^{1,\text{st}} \qquad (4.b)$$

195

$$\sum_{t' \leq t} \sum_{h \in H_{s,t'}^{\bar{\Omega}2+}} x_h^2 - \sum_{t' \leq t} \sum_{h \in H_{s,t'}^{\bar{\Omega}2-}} x_h^2 \leq 0 \qquad \forall s \in S, t \in \mathbb{T}_s^{2,\text{st}} \qquad (4.\text{c})$$

$$\sum_{t \in \mathbb{T}} \sum_{h \in H_{o^2,t}^{\bar{\Omega}2+}} x_h^2 = \sum_{t \in \mathbb{T}} \sum_{h \in H_{o^2,t}^{\bar{\Omega}2-}} x_h^2 = F^2 \qquad (4.\text{d})$$

$$\sum_{\substack{t' \leq t-p_s \\ t' \in \mathbb{T}_s^{1,\text{vis}}}} \left(\sum_{h \in \Omega^1} \gamma_{h,s,t'}^{\text{fw}} x_h^1 - u_{s,t'}^{\text{fw}} \right) - \sum_{t' \leq t} \sum_{h \in \Omega^2} \gamma_{h,s,t'}^{\text{fw}} x_h^2 \geq 0 \qquad \forall s \in S, t \in \mathbb{T}_s^{2,\text{st}} \qquad (4.\text{e})$$

$$\sum_{\substack{t' \geq t+p_s \\ t' \in \mathbb{T}_s^{1,\text{vis}}}} \left(\sum_{h \in \Omega^1} \gamma_{h,s,t'}^{\text{rv}} x_h^1 - u_{s,t'}^{\text{rv}} \right) - \sum_{t' \geq t} \sum_{h \in \Omega^2} \gamma_{h,s,t'}^{\text{rv}} x_h^2 \geq 0 \qquad \forall s \in S, t \in \mathbb{T}_s^{2,\text{end}} \qquad (4.\text{f})$$

$$\sum_{\substack{t' \leq t \\ t' \in \mathbb{T}_s^{1,\text{vis}}}} \left(\sum_{h \in \Omega^1} \gamma_{h,s,t'}^{\text{fw}} x_h^1 - u_{s,t'}^{\text{fw}} \right) - \sum_{t' \leq t} \sum_{h \in \Omega^2} \gamma_{h,s,t'}^{\text{fw}} x_h^2 +$$

$$\sum_{\substack{t' \geq t \\ t' \in \mathbb{T}_s^{1,\text{vis}}}} \left(\sum_{h \in \Omega^1} \gamma_{h,s,t'}^{\text{rv}} x_h^1 - u_{s,t'}^{\text{rv}} \right) - \sum_{t' \geq t} \sum_{h \in \Omega^2} \gamma_{h,s,t'}^{\text{rv}} x_h^2 \leq C_s^{sat} \qquad \forall s \in S, t \in \mathbb{T}_s^{1,\text{vis}} \qquad (4.\text{g})$$

$$\sum_{h \in \Omega^1} \gamma_{h,s,t}^{\text{fw}} x_h^1 - u_{s,t}^{\text{fw}} \geq 0 \qquad \forall s \in S, t \in \mathbb{T}_s^{1,\text{vis}} \qquad (4.\text{h})$$

$$\sum_{h \in \Omega^1} \gamma_{h,s,t}^{\text{rv}} x_h^1 - u_{s,t}^{\text{rv}} \geq 0 \qquad \forall s \in S, t \in \mathbb{T}_s^{1,\text{vis}} \qquad (4.\text{i})$$

$$\sum_{h \in \omega_s^1} x_h^1 + \sum_{h \in \omega_s^{2+} \cup \omega_s^{2-}} x_h^2 \leqslant M.\zeta_s \qquad \forall s \in S \qquad (4.\text{j})$$

$$x_h^1 \in \mathbb{N}_0 \qquad \forall h \in \bar{\Omega}^1$$

$$x_h^2 \in \{0,1\} \qquad \forall h \in \bar{\Omega}^2$$

$$u_{s,t}^{\text{fw}}, u_{s,t}^{\text{rv}} \geq 0 \qquad \forall s \in S, t \in \mathbb{T}_s^{1,\text{vis}}$$

$$\sigma_i \geq 0 \qquad \forall i \in N$$

$$\zeta_s \in \{0,1\} \qquad \forall s \in S$$

The objective integrate to the one of Model 3, the new penalty variables with Big-M coefficients for unserved customer identified with σ variables. Constraints (4.a) states that a customer is either served by a second-echelon trip or the penalty has to be paid, through the σ variables. Fleet-size Constraints (4.b)–(4.d) are identical to Formulation 3 except for the use of sets $\bar{\Omega}^1$ and $\bar{\Omega}^2$. Constraints (4.e)-(4.g) are the same flow balancing and capacity equations as in Model 3, with the subtraction the new variables u to manage the surplus quantities. Constraints (4.h) and (4.i) limit the variables u to the quantity dropped-off (collected) at satellite s at time t by first-echelon trips. In addition, the constraints (4.j) allow trips to visit a satellite only if the satellite is utilized, according to the new ζ variables. Finally, the variable domains are given.

DETAILED RESULTS ON 2-ECHELON VEHICLE ROUTING PROBLEMS

C.1 Detailed results on the 2E-VRPTW instances

Instance name	Cost	Instance name	Cost	Instance name	Cost
c101	471.03	r101	872.37	rc101	802.84
c102	459.75	r102	800.18	rc102	757.23
c103	443.16	r103	703.95	rc103	743.34
c104	435.11	r104	666.01	rc104	709.95
c105	471.03	r105	801.67	rc105	828.21
c106	471.03	r106	714.36	rc106	759.81
c107	465.75	r107	673.67	rc107	718.17
c108	461.93	r108	652.33	rc108	700.96
c109	440.13	r109	701.08	rc201	619.93
c201	455.81	r110	695.84	rc202	592.38
c202	445.23	r111	677.73	rc203	607.06
c203	441.62	r112	648.10	rc204	595.39
c204	438.10	r201	717.33	rc205	564.54
c205	453.27	r202	661.68	rc206	579.80
c206	453.27	r203	641.77	rc207	576.01
c207	449.68	r205	648.51	rc208	524.35
c208	445.70	r206	659.74		
		r207	642.39		
		r208	574.47		
		r210	651.19		

Table C.1 – Performance on the 2E-VRPTW instances with 25 customers compared to the BPC algorithm of Marques et al. (2020a)

Instance name	Cost	Instance name	Cost	Instance name	Cost
c101	1 008.91	r101	1 316.38	rc101	1 505.31
c102	978.24	r102	1 185.01	rc102	1 429.51
c103	947.95	r103	1 086.47	rc103	1 383.80
c104	916.21	r104	963.87	rc104	1 265.18
c105	995.86	r105	1 198.87	rc105	1 487.50
c106	1 011.95	r106	1 080.09	rc106	1 453.06
c107	975.37	r107	1 033.93	rc107	1 431.03
c108	968.11	r108	937.21	rc108	1 315.12
c109	934.44	r109	1 065.14	rc201	968.99
c201	744.20	r110	1 036.03	rc202	916.46
c202	740.53	r111	993.24	rc203	866.15
c203	759.51	r112	1 002.22	rc205	934.23
c204	706.63	r201	1 064.08	rc206	911.71
c205	741.87			rc207	890.79
c206	741.32				
c207	754.60				
c208	727.20				

Table C.2 – Performance on the 2E-VRPTW instances with 50 customers compared to the BPC algorithm of Marques et al. (2020a)

Instance name	Cost	Instance name	Cost	Instance name	Cost
c101	1 480.58	r101	2 033.36	rc101	2 019.64
c102	1 434.96	r102	1 863.84	rc102	1 915.65
c103	1 443.70	r103	1 604.68	rc103	1 790.81
c104	1 309.81	r104	1 428.00	rc105	1 959.16
c105	1 431.34	r105	1 792.73	rc106	1 879.94
c106	1 455.36	r106	1 707.32	rc107	1 795.84
c107	1 403.53	r107	1 537.08	rc108	1 731.54
c108	1 393.20	r110	1 608.41		
c201	985.71	r111	1 522.28		
c202	1 007.61	r112	1 539.07		
c205	946.99				
c206	944.77				

Table C.3 – Performance on the 2E-VRPTW instances with 75 customers compared to the BPC algorithm of Marques et al. (2020a)

Instance name	Cost	Instance name	Cost	Instance name	Cost
c101	2 007.48	r101	2 302.51	rc101	2 548.11
c102	1 933.68	r102	2 131.33	rc102	2 403.70
c105	1 911.79	r105	2 064.17	rc105	2 549.57
c106	1 920.90			rc106	2 375.75
c107	1 864.91			rc106	2 375.75
c108	1 867.74				
c109	1 939.84				
c201	1 283.63				

Table C.4 – Performance on the 2E-VRPTW instances with 100 customers compared to the BPC algorithm of Marques et al. (2020a)

C.2 Detailed results on the 2E-VRPTW-CSRF instances

Instance name	Capa 20	Capa 25	Capa 35	Capa 50	Capa 70	Capa 500
A-50-1	1 876	1 759	1 642	1 598	1 598	1 598
A-50-2	-	1 956	1 907	1 847	1 847	1 842
A-50-3	1 976	1 870	1 798	1 753	1 753	1 753
A-50-4	-	2 042	1 914	1 862	1 862	1 862
A-50-5	-	2 260	2 116	2 081	2 066	2 066
A-50-6	1 993	1 988	1 973	1 973	1 973	1 973
A-50-7	1 988	1 905	1 819	1 818	1 798	1 738
A-50-8	-	1 854	1 854	1 791	1 727	1 727
A-50-9	-	2 170	2 030	1 986	1 936	1 936
A-50-10	1 931	1 846	1 776	1 714	1 700	1 700
B-50-1	1 960	1 881	1 835	1 835	1 802	1 802
B-50-2	1 857	1 857	1 787	1 749	1 749	1 749
B-50-3	1 888	1 733	1 713	1 713	1 713	1 699
B-50-4	1 765	1 647	1 641	1 566	1 566	1 566
B-50-5	1 768	1 721	1 713	1 712	1 660	1 641
B-50-6	1 762	1 684	1 664	1 664	1 616	1 616
B-50-7	1 625	1 573	1 533	1 533	1 521	1 521
B-50-8	1 875	1 822	1 752	1 752	1 752	1 746
B-50-9	1 760	1 738	1 710	1 710	1 710	1 710
B-50-10	1 661	1 601	1 585	1 566	1 566	1 547
C-50-1	1 746	1 666	1 666	1 666	1 666	1 666
C-50-2	1 772	1 762	1 733	1 677	1 631	1 631
C-50-3	1 637	1 571	1 546	1 546	1 513	1 513
C-50-4	1 845	1 732	1 654	1 654	1 654	1 654
C-50-5	1 800	1 800	1 711	1 689	1 689	1 624
C-50-6	1 783	1 781	1 703	1 703	1 703	1 700
C-50-7	1 553	1 449	1 449	1 449	1 449	1 449
C-50-8	1 815	1 770	1 692	1 636	1 636	1 636
C-50-9	1 578	1 578	1 578	1 542	1 542	1 534
C-50-10	1 555	1 522	1 487	1 487	1 487	1 487

Table C.5 – Results on the 2E-MTVRP-CSRF instances with 50 customers

Instance name	Capa 20	Capa 25	Capa 35	Capa 50	Capa 70	Capa 500
A-50-1	-	3 384	3 110	2 992	2 821	2 782
A-50-2	-	3 756	3 525	3 273	3 180	3 180
A-50-3	-	3 167	2 840	2 744	2 671	2 660
A-50-4	3 921	3 685	3 319	3 086	3 086	3 086
A-50-5	3 789	3 338	2 999	2 822	2 783	2 783
A-50-6	-	-	3 380	3 187	3 093	3 036
A-50-7	-	3 588	3 214	3 027	2 956	2 935
A-50-8	3 835	3 354	3 128	2 870	2 813	2 813
A-50-9	3 682	3 380	3 196	2 921	2 869	2 848
A-50-10	3 279	2 967	2 727	2 637	2 566	2 566
B-50-1	3 313	3 157	2 932	2 769	2 669	2 669
B-50-2	3 490	3 358	3 112	3 041	2 865	2 865
B-50-3	3 268	3 096	2 934	2 831	2 767	2 767
B-50-4	3 064	2 952	2 816	2 605	2 605	2 605
B-50-5	3 164	3 124	2 923	2 846	2 728	2 728
B-50-6	3 073	2 837	2 547	2 547	2 516	2 516
B-50-7	3 660	3 464	3 235	3 083	2 961	2 933
B-50-8	2 541	2 526	2 412	2 391	2 391	2 373
B-50-9	3 344	3 166	2 933	2 745	2 667	2 667
B-50-10	2 702	2 538	2 423	2 423	2 406	2 406
C-50-1	2 948	2 831	2 631	2 531	2 524	2 514
C-50-2	2 686	2 546	2 483	2 410	2 363	2 326
C-50-3	2 811	2 565	2 485	2 485	2 473	2 456
C-50-4	2 704	2 508	2 429	2 408	2 370	2 370
C-50-5	2 476	2 476	2 422	2 367	2 367	2 367
C-50-6	2 554	2 513	2 462	2 354	2 354	2 354
C-50-7	2 449	2 449	2 340	2 340	2 279	2 279
C-50-8	2 732	2 707	2 629	2 479	2 479	2 477
C-50-9	2 383	2 326	2 287	2 265	2 265	2 265
C-50-10	2 895	2 799	2 659	2 593	2 593	2 593

Table C.6 – Results on the 2E-MTVRP-CSRF instances with 100 customers

LIST OF FIGURES

204

LIST OF TABLES

Titre : Approches matheuristiques pour la résolution de problèmes d'optimisation des transports en logistique urbaine

Mot clés : logistique urbaine, tournée de véhicules, matheuristique, recherche à voisinage large, ressources synchronisées

Résumé : Dans cette thèse, nous nous intéressons aux problématiques soulevées par l'optimisation des tournées de véhicules pour la livraison en milieu urbain. Nous nous sommes attachés à la problématique des ressources synchronisées. Le contexte urbain impose des contraintes sur l'espace utilisable. Dans les modèles théoriques développés, cela se traduit par des contraintes de ressources communes à tous les véhicules.

Le premier problème que nous avons développé traite de la livraison de colis avec fenêtres horaire en considérant de multiples options de livraison pour chaque commande. La synchronisation de ressources vient de la prise en compte d'un niveau de service global ainsi que de la capacité de lieux de livraison partagé, tels que des consignes. Le second problème traité vise à optimiser la collecte et la livraison de colis via un système logistique à deux échelons. Les ressources synchronisées sont alors la capacité de stockage des entrepôts intermédiaires, appelé satellites.

Pour résoudre ces problèmes, nous avons développé des méthodes de recherche à voisinage large basé sur de petites destructions. Nous avons aussi étudié leur hybridation avec la résolution de modèles MIP et de la programmation dynamique. Ainsi, nos méthodes sont catégorisées comme matheuristiques.

Title: Matheuristic approaches for solving transport optimization problems in urban logistics

Keywords: city logistics, vehicle routing, matheuristic, large neighborhood search, ressources synchronizations

Abstract: In this thesis, we are interested in the problems raised by the optimization of vehicle routing for delivery in urban areas. We pay close attention to problems with synchronized resources. Indeed, the urban context imposes hard constraints on the usable space. In the theoretical models developed, this translates into resource constraints common to all vehicles.

The first problem deals with the delivery of parcels with time windows while considering multiple delivery options for each order. The synchronization of resources between vehicles comes from the consideration of the global quality of service according to the customers' preferences as well as the capacity of shared delivery locations, such as lockers. The second problem addresses the optimization of the collection and delivery of parcels via a two-echelon logistics system. The synchronized resources between all vehicles are then the limited storage capacity of intermediate warehouses, called satellites.

To solve these problems, we develop a large neighborhood search methods based on small destructions. We also study their hybridization with MIP models and dynamic programming. Thus, our methods are categorized as matheuristics.